Titre: Title:	Codage Turbo avec techniques de modulations multi-niveaux et en treillis
Auteur: Author:	Isabelle Yehouessi Metoevi
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Metoevi, I. Y. (2004). Codage Turbo avec techniques de modulations multi-niveaux et en treillis [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7504/

NOTE TO USERS

UNIVERSITÉ DE MONTRÉAL

CODAGE TURBO AVEC TECHNIQUES DE MODULATIONS MULTI-NIVEAUX ET EN TREILLIS

ISABELLE YEHOUESSI METOEVI
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)
(GÉNIE ÉLECTRIQUE)
JUILLET 2004

Your file Votre référence
ISBN: 0-612-97969-5
Our file Notre référence
ISBN: 0-612-97969-5

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CODAGE TURBO AVEC TECHNIQUES DE MODULATIONS MULTI-NIVEAUX ET EN TREILLIS

présenté par: METOEVI Isabelle Yehouessi

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. CARDINAL Christian, Ph.D., président

M. HACCOUN David, Ph.D., membre et directeur de recherche

M. NERGUIZIAN Chahé, Ph.D., membre

À mes parents

REMERCIEMENTS

Je tiens tout d'abord à exprimer ma reconnaissance envers mon directeur de recherche, le professeur David Haccoun pour son assistance constante tout au long de mon projet et également pour l'aide financière qu'il m'a accordée pendant mes recherches.

Je remercie mon Dieu qui m'a permis de commencer ces études et de les terminer, et qui m'a assisté pendant mes études. Un grand merci également à mes parents qui m'ont permis d'entreprendre ces études, pour leurs prières et pour leur soutien financier tout au long de mes études universitaires.

J'ai également une pensée pour tous mes collègues du laboratoire de communications, Yucheng, Aniss, Ahmad, Patrick, Dan, Mahdi, Oualid, Laynish, Zouheir, Boubaker et Eric pour leurs précieux conseils.

RÉSUMÉ

Dans les systèmes de transmissions numériques, le codage de canal est utilisé pour corriger les erreurs survenues lors de la transmission du signal dans le canal bruité. Parmi les techniques de codage actuelles, les codes turbo proposés par Berrou, Glavieux et Thitimajshima sont les plus puissants, leurs performances en terme de probabilité d'erreur se rapprochent de très près de la limite théorique de Shannon (à 0.7 db). Malheureusement, la modulation BPSK avec laquelle ces codes ont été introduits n'offre pas une très bonne efficacité spectrale. Dans le but d'améliorer cette efficacité spectrale et d'augmenter la vitesse de transmission il serait avantageux de combiner les codes turbo avec les modulations à grande efficacité spectrale telles que le M-QAM et le M-PSK.

Ce mémoire porte essentiellement sur les techniques de modulations élévées associées aux codes turbo dans un canal à bruit blanc gaussien. Nous faisons une étude principalement sur deux schémas de modulation, le TTCM et le Turbo pragmatique. Leurs performances d'erreurs sont évaluées par simulations à l'ordinateur et analysées en fonction de la longueur de contrainte et d'entrelaceur, du type de modulation et de l'efficacité spectrale.

Nous avons étudié l'influence du nombre d'itération sur les performances d'erreur. Plus le nombre d'itérations augmente, meilleures sont ces performances. Malheureusement cette amélioration ne va pas sans une complexité supplémentaire. L'itération 6 nous donne le meilleur compromis entre la complexité et la performance d'erreur aussi bien pour le schéma TTCM que le schéma turbo pragmatique.

Au niveau du schéma turbo pragmatique nous avons étudié l'influence de l'assignation des bits à la constellation sur les performances d'erreurs. Pour une probabilité d'erreur supérieure à $5 * 10^{-5}$ et inférieure à $5 * 10^{-5}$, les assignations S_2 et S_1 respectivement présentent les meilleures performances d'erreur. S_3 présente des résultats intermédiaires. S_2 privilégie les bits de parité sur les deux dimensions, S_1 les bits d'information sur les deux dimensions et S_3 privilégie un

bit d'information sur une dimension et un bit de parité sur l'autre dimension. Ces résultats sont valides pour une longueur de contrainte $K = 4$. Pour $K = 5$ par contre, l'assignation S_2 sera toujours la meilleure. Il apparaît aussi que l'influence de l'assignation diminue nettement avec l'augmentation de l'efficacité spectrale pour un même type de constellation. Pour une efficacité spectrale égale à 5 bits/s/Hz, il n'y a plus aucune différence entre les assignations.

Au niveau de l'influence de la longueur de contrainte sur les performances d'erreur, nous avons trouvé les mêmes caractéristiques aussi bien pour le schéma TTCM que pour le turbo pragmatique. Le code de longueur de contrainte $K = 3$ ne dispose pas d'une capacité de correction suffisante pour présenter un intérêt. Seules les longueurs de contrainte 4 et 5 présentent un excellent compromis entre la performance d'erreur et la complexité. Avec $K = 5$ nous obtenons un léger gain, de 0.6 à 0.8 dB par rapport à $K = 4$ pour une probabilité d'erreur 10^{-5}. Au delà de $K = 5$, pour $K = 6$ et $K = 7$, on n'a aucune amélioration de la performance d'erreur, ces deux longueurs de contrainte ne présentent donc aucun interêt.

La longueur de l'entrelaceur influence également énormement les performances des codeurs turbo pragmatique et TTCM. On peut avoir des gains de codage de 2 dB en passant d'une longueur d'entrelaceur de 256 à 4096 pour une probabilité d'erreur de 10^{-5}. Malheureusement cela implique une augmentation du délai.

Des simulations ont été faites également pour voir l'influence de l'assignation selon Gray ou selon Ungerboeck sur les performances du codeur TTCM, il s'est avéré que l'assignation selon Ungerboeck est de loin la meilleure assignation, on va jusqu'à un gain de codage de 3.7dB d'une assignation à une autre.

Pour finir le mémoire nous avons fait une étude comparative des schémas de modulation TTCM et turbo pragmatique. Les performances d'erreur obtenues sont les mêmes. Le schéma turbo pragmatique a l'avantage d'être modulaire mais son algorithme de décodage est assez complexe ce qui n'est pas le cas pour le schéma TTCM mais qui présente l'inconvénient de n'être pas modulaire.

ABSTRACT

In digital communications, channel coding is applied for the correction of errors that appear during the transmission of symbols via an Additive White Gaussian Noise (AWGN) channel. Among the recently discussed coding techniques, the turbo codes which were introduced by Berrou, Glavieux and Thitimajshima are the most powerful in the sense of the bit error probability as they enable us to approach the Shannon limit within 0.7 dB. However, these codes, originally introduced for BPSK modulation, suffer from a poor spectral efficiency. For an increase of this bandwidth efficiency and thus the transmission speed, it is attractive to combine turbo codes with high-order modulation schemes such as M-QAM and M-PSK.

This master thesis addresses essentially high-order modulation schemes for turbo coding and a Gaussian channel noise model. We consider two modulation schemess in detail: TTCM (Turbo Trellis Coded Modulation) and pragmatic turbo coding. The error performance is evaluated by computer simulation and analyzed as a function of the length of decoding registers and the interleaver, the modulation scheme and the bandwidth efficiency.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET DES SYMBOLES

Les sigles

AVA	Adaptative Viterbi Algorithm
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BCJR	Bahl Cocke Jelinek Raviv algorithm
BPSK	Binary Phase Shift Keying
CE	Cross Entropy
CRS	Convolutionnel Récursif Systématique
DEC1	Décodeur 1
DEC2	Décodeur 2
FSK	Frequency Shift Keying
HDA	Hard Decision Aided
LLR	Log Likelihood Ratio
MAP	Maximun à Posteriori
PAM	Pulse Amplitude Modulation
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
SCR	Sign Change Ratio
SOVA	Soft Output Viterbi Algorithm
TCM	Trellis Coded Modulation
T-TCM	Turbo Trellis Coded Modulation

Les symboles

$A_{w,z}^{C}$	Fonction de distribution de poids d'un codeur turbo
$C^2(i)$	Nombre de changement de signe entre les éléments des séquences d'informations extrinsèques
$d_c(n)$	Fonction de distance des colonnes d'ordre n d'un code convolutionnel
d_{free}	Distance libre d'un code
d_h	Distance de Hamming
D	Unité de décalage pour les entrelaceurs convolutionnels
$\boldsymbol{E}$	Opérateur défini par $x\boldsymbol{E}y = -ln(e^{-x} + e^{-y})$
E_b	Énergie du signal par bit
E_b/N_0	Rapport signal sur bruit
E_s	Énergie du signal par symbole
$\tilde{E}_s$	Énergie moyenne du signal par symbole
$\boldsymbol{G}$	Matrice génératrice d'un code
K	Longueur de contrainte d'un code convolutionnel
La_k	Information a priori à l'instant k
L_e	Information extrinsèque
M	Mémoire d'un code convolutionnel
N	Taille d'une séquence d'information binaire
N_0	Densité spectrale de puissance de bruit
P	Matrice de perforation
P_B	Probabilité d'erreur par bit
$Q()$	Fonction Q
R	Taux de codage d'un code convolutionnel
R_g	Taux de codage global
R_k	Symbole reçu à l'instant k

S_k	État du codeur à l'instant k
$T(D, B, L)$	Fonction de transfert d'un code convolutionnel
V	Nombre d'additionneurs modulo 2 d'un code convolutionnel
X_k	Symbole codé émis dans le canal à l'instant k
X_k^{ip}	Symbole de parité du codeur i du codeur turbo
X_k^s	Symbole systématique à la sortie de l'encodeur turbo
Y_k^s	Symbole d'information systématique reçu à l'instant k
Y_k^{1p}	Symbole de parité 1 reçu à l'instant k
Y_k^{2p}	Symbole de parité 2 reçu à l'instant k
$\alpha_k(m)$	Métrique d'état en avant à l'instant k
$\beta_k(m')$	Métrique d'état en arrière à l'instant k
$\gamma_i(R_k, m', m)$	Métrique de branche à l'instant k
σ^2	Variance du bruit dans le canal

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

Le but principal d'une communication numérique est de transmettre de l'information avec un minimun d'erreur. Cependant cette transmission est, la plupart du temps, pertubée par du bruit provenant du canal. Ceci entraîne une dégradation de la fiabilité du message. Une des solutions pour détecter et corriger les erreurs serait d'augmenter la puissance d'émission, mais cette solution est souvent très coûteuse. Une autre solution efficace est de coder l'information. Le codage consiste à ajouter de l'information redondante selon des règles bien précises à l'information initiale à émettre. Cette information redondante a pour but de protéger le signal contre les erreurs de transmission, car l'algorithme de décodage utilise cette redondance pour détecter et eventuellement corriger les symboles érronés.

En 1948, Shannon a démontré qu'il est théoriquement possible de transmettre de l'information sans aucune erreur en utilisant un codage de canal approprié, à condition que le débit de transmission de la source R_s, soit inférieur à la capacité du canal C. Il a montré que pour un canal à bruit gaussien blanc additif de densité de puissance $No/2$, la capacité C est donnée par:

$$C = Wlog_2(1 + \frac{P}{NoW}) \ \ bits/s \tag{1.1}$$

où W est la largeur de bande du canal exprimée en Hertz et P, la puissance moyenne du signal. En principe, cela signifie que si on abandonne la transmission non codée au profit d'une technique avec codage, nous pourrions effectuer une transmission sans erreurs. Malheureusement, le théorème proposé par Shannon, ne fournit aucune méthode pratique pour déterminer un tel code.

Le processus de codage dans un système de communication numérique s'effectue en deux étapes : le codage de source et le codage de canal. Le rôle du codeur de source est d'éliminer la redondance contenue dans l'information produite par la source et de représenter celle-ci numériquement alors que le codeur de canal adapte le signal au canal et réintroduit généralement une redondance contrôlée sous la forme de symboles de parité.

C'est le domaine spatial qui a permis le développement de ces codes correcteurs d'erreurs. En effet les premières utilisations de ceux-ci devaient permettre de surmonter la perte de puissance lors de la propagation du signal entre l'émetteur et le recepteur. Le but des codes correcteurs d'erreurs était de gagner en terme de puissance c'est à dire d'avoir la plus petite probabilité d'erreur en utilisant un minimum de puissance d'émission.

Il existe deux grandes familles des codes correcteurs d'erreurs (codeur de canal). La première est celle des codes en blocs où une séquence de k bits d'information est codée dans un bloc de n symboles ($n > k$). La deuxième est celle des codes convolutionnels, où les symboles d'information sont codés de façon continue contrairement aux codes en blocs où il faut attendre que tous les k bits se retrouvent à l'entrée pour effectuer le codage. Dans le codage convolutionnel les bits d'entrée passent dans un registre à décalage ce qui assure une continuité de leur codage. Dans nos travaux, la technique de codage utilisée est le codage convolutionnel.

Le but de la recherche dans le domaine de la théorie de l'information, est de trouver un système de contrôle d'erreur qui s'approchera le plus de la limite de Shannon. Pour cette raison, il faut trouver des codes ayant de bonnes propriétés de distance, déterminer leurs structures, et développer un algorithme de décodage adéquat ayant la plus faible complexité possible.

Actuellement, la technique de codage la plus puissante est la technique de codage turbo présentée en 1993 par Berrou, Glavieux et Thitimajshima [7]. Les

performances obtenues sont exceptionnelles car elles s'approchent à 0.7 dB de la limite de Shannon [7]. Mais de telles performances ne sont obtenues qu'au prix d'un délai et d'une complexité importants surtout au niveau de l'algorithme de décodage.

Toujours dans le but d'améliorer les limites des codes quant à l'efficacité spectrale, nous avons essentiellement etudié les techniques de modulations élevées, sous un canal gaussien pour des longueurs de contrainte différentes, une efficacité spectrale différente et des modulations d'ordre élevé allant du 16 QAM au 64 QAM.

1.1 Composition du mémoire

Ce mémoire est structuré de la manière suivante :

Le chapitre 1 présente l'importance du codage de canal dans une communication numérique et les différents types de codage qui existent dans la littérature.

Le chapitre 2 présente de manière générale un système de communication numérique suivi d'une présentation des codes convolutionnels systématiques récursifs et non récursifs. Le chapitre se termine par une description de la modulation BPSK et des différents types de canaux.

Le chapitre 3 est une présentation de la structure des codes turbo et de ses paramètres importants. Nous présentons d'abord le principe général des codes turbo, ensuite les différents types d'entrelaceurs et pour finir, un élément important des codes turbo : le décodage itératif. A cet effet nous allons présenter deux algorithmes importants de décodage à savoir l'algorithme MAP et l'algorithme Log-MAP.

Le chapitre 4 est consacré aux modulations multi-niveaux associées aux codes turbo. Nous présentons d'abord brièvement les différents schémas qui existent dans la littérature, puis ensuite nous étudions notamment le codage turbo pragmatique associé aux différentes modulations telles que 16-QAM, 32-QAM et 64-QAM.

L'étude comprend également l'analyse de l'influence de la longueur de contrainte des codes, de la taille des entrelaceurs et de l'efficacité spectrale. Une étude est faite également sur l'influence de l'assignation des bits à la constellation dans les modulations multi-niveaux.

Le chapitre 5 est consacré à l'étude des modulations turbo codées en treillis(TTCM), et finit par une étude comparative basée sur des simulations, des schémas TTCM et turbo pragmatique.

Le dernier chapitre, le chapitre 6 fait l'objet de la conclusion du mémoire. Nous résumons les cinq chapitres qui le précédent et parlons des résultats importants qui s'y sont dégagés. Le chapitre se termine par une proposition de nouveaux champs de recherches.

1.2 Contributions

Les contributions apportées par ce travail de recherche sont les suivantes :

- Etude par simulation du décodage turbo pragmatique pour une efficacité spectrale allant de 2 à 7 bits/s/Hz et pour les constellations 8-PSK, 16-QAM, 32-QAM, 64-QAM, 128 QAM et 256-QAM.
- Etude de l'influence de l'assignation des bits à la constellation et de l'influence de l'efficacité spectrale sur l'assignation des bits à la constellation.
- Etude de la longueur de contrainte des codes sur les performances des schémas turbo pragmatique et TTCM.
- Etude comparative des schémas turbo pragmatique et TTCM.
- Programmation en langage C du schéma TTCM, étude des performances d'erreur du schéma TTCM pour des longueurs de contrainte différentes, pour les modulations 8-PSK, 16-PSK et 16-QAM.

- Etude comparative pour le schéma TTCM du codage de Gray et du partitionnement de Ungerboeck.

CHAPITRE 2

CODES CONVOLUTIONNELS

2.1 Systèmes de communications numériques

Un système de communication numérique se présente géneralement comme le montre la figure 2.1. La première opération sera d'effectuer un codage de source. Ce dernier est principalement utilisé pour réduire la redondance dans la séquence d'information, il consiste également à convertir la source de données qui la plupart du temps se présente sous la forme analogique en une séquence binaire. Dans ce mémoire nous nous intéresserons uniquement au codage de canal.

Une fois le codage de source effectué, nous devons passer à l'étape codage de canal. Cette dernière étape a pour but d'ajouter de la redondance contrôlée au message de façon à pouvoir détecter et/ou corriger les erreurs intervenues lors de la transmission de l'information dans le canal bruité. La dernière étape consiste à moduler le signal. Le modulateur joue le rôle d'adaptateur de canal. Son rôle est de faire en sorte que la séquence soit adaptée à sa transmission dans le canal physique dans lequel le signal est transmis. La source de données ainsi modifiée passe à travers d'un canal physique.

A la sortie du canal, pour retrouver le signal origine il faut effectuer les étapes inverses de celles que nous avons décrites plus haut. Le signal est donc tout d'abord démodulé pour être ensuite décodé par le décodeur de canal. Le décodage est un algorithme adapté au codage d'origine et qui permet de retrouver le signal codé en minimisant la probabilité d'erreur. Enfin, le décodage de source terminera le processus.

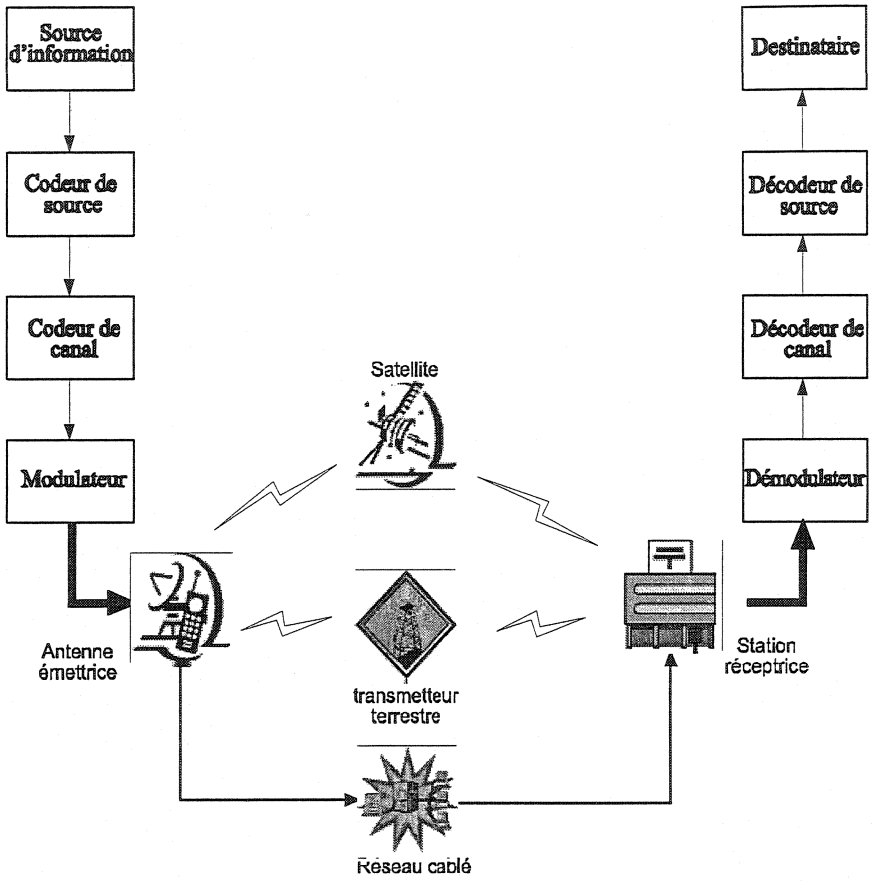

Figure 2.1: Représentation générale d'un système de communication numérique

2.2 Les codes convolutionnels

2.2.1 Principe de base

Le codeur convolutionnel de taux $R = 1/V$ et de longueur de contrainte K, introduit par Elias [18] est une machine linéaire à états finis constituée d'un registre à décalage de K cellules, de V additionneurs modulo 2 qui sont reliés à certaines des cellules du registre et d'un commutateur qui sélectionne séquentiellement les sorties des V additionneurs. La valeur K est appelée la longueur de contrainte du codeur. Un exemple de codeur est representé à la figure 2.2.

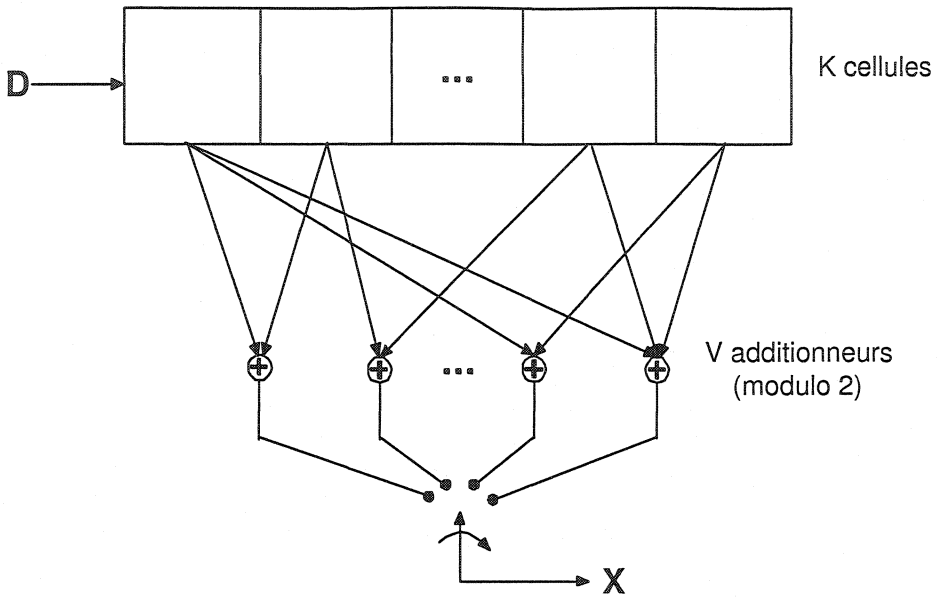

Figure 2.2: Codeur convolutionnel de taux de codage $R = 1/V$

Supposons qu'au début les K cellules du registre à décalage soient initialisées à zéro. Les bits constituant la séquence de bits d'information de longueur N sont introduits un par un dans le registre. Après l'introduction de chaque bit, les V additionneurs modulo-2 sont balayés séquentiellement par le commutateur afin de fournir les V symboles codés : $x_i^{(j)}$, $i = 1, 2, ...N$ et $j = 1, 2, ..., V$. Le contenu des cellules est ensuite déplacé d'un bit vers la droite, afin que le deuxième bit d'information soit introduit dans la première cellule, fournissant ainsi les nouveaux V symboles codés et ainsi de suite jusqu'à l'entrée du dernier bit. La réinitialisation du codeur se fait ensuite par l'envoi de $(K-1)$ zéros. Cette sequence de zéros est appelée la queue du message. Pour mieux illustrer les codes convolutionnels nous prendrons l'exemple du codeur convolutionnel montré à la figure 2.3.

Pour un bit d'information d_k donné, nous avons l'expression de la séquence codée $x_k^{(j)}$:

$$x_k^{(j)} = \sum_{i=0}^{K-1} \oplus \; g_{ji} \; d_{(k-i)}, \quad j = 1, \ldots, V., \quad k = 1, \ldots, N. \tag{2.1}$$

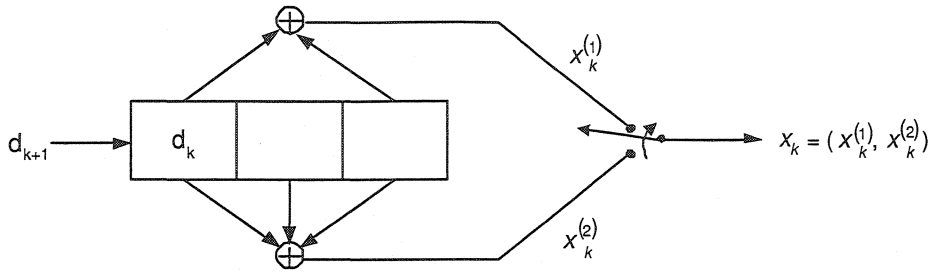

Figure 2.3: Codeur convolutionnel de taux de codage $R = 1/2, K = 3, V = 2$

où g_{ij}, de valeur 0 ou 1, indique la présence ou non de connexion entre le $i^{\text{ième}}$ additionneur et la $j^{\text{ième}}$ cellule du registre à décalage.

Dans notre exemple de la figure 2.3 nous avons deux générateurs $\mathbf{G_1} = (101)$ et $\mathbf{G_2} = (111)$ relatifs aux deux additionneurs. En général on exprime les générateurs en octal ce qui donne $\mathbf{G_1} = 5$ et $\mathbf{G_2} = 7$.

De l'équation 2.1 on en déduit que:

$$\mathbf{X} = \mathbf{D}\ [\mathbf{G}] \tag{2.2}$$

$\mathbf{X}$ étant la représentation matricielle du signal de sortie, $\mathbf{D}$ celle de la séquence d'information et $[\mathbf{G}]$, la matrice génératrice du codeur convolutionnel, qui peut également se déterminer par la réponse impulsionnelle du codeur. La réponse impulsionnelle correspond à la sortie pour une entrée 100000....

2.2.2 Représentations graphiques des codes convolutionnels

Il existe trois représentations graphiques des codes convolutionnels à savoir le diagramme d'états, la représentation en arbre et la représentation en treillis.

- Le diagramme d'états

  Le diagramme d'état est un graphe décrivant les transitions d'état du codeur.

L'état d'un codeur convolutionnel correspond à la valeur que prend les cellules du registre d'états à un instant donné. Le diagramme d'états est constitué de 2^{K-1} sommets correspondant aux différents états du codeur et d'arcs orientés représentants les transitions entre les différents états. Nous étiquetons les flèches à l'aide du doublet (bit d'information à l'entrée du codeur, symboles codés à la sortie du codeur).

Le diagramme d'états est une représentation efficace des transitions d'un codeur, il est très utilisé pour les codeurs dont le nombre d'états est assez petit. Le diagramme d'états du codeur de la figure 2.3 est représenté à la figure 2.4.

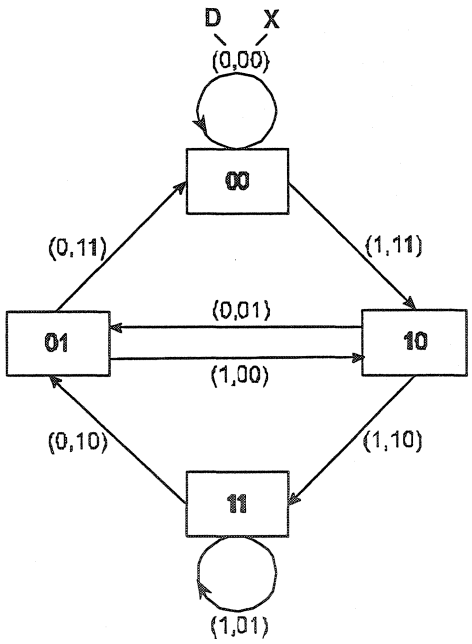

Figure 2.4: Diagramme d'états du codeur de la figure 2.3

Le diagramme d'états nous permet de déterminer la fonction de transfert du code. Il est utilisé pour évaluer la distribution des poids de Hamming des différents chemins qui divergent de l'état initial (état nul) et qui reconvergent

plus loin. Le diagramme d'états est un outil pratique pour caractériser les codes mais il ne permet cependant pas de suivre aisément les évolutions dynamiques du codeur dans le temps. Pour pallier à cet inconvénient, deux repésentations schématiques plus complètes du codeur sont alors utilisées : la représentation en arbre et la représentation en treillis. Ces deux représentations fournissent une information temporelle qui n'est pas disponible avec le diagramme d'état.

- La représentation en arbre

  Dans cette représentation, les transitions entre les états de codeur sont illustrées comme un cheminement à partir d'un état de départ (généralement l'état zéro) qui est l'origine de l'arbre, vers les états ultérieurs, qui sont reliés par les branches de l'arbre.

  Chaque nœud de l'arbre représente un état du codeur, la branche supérieure correspond à un bit d'information "0" et la branche inférieure à un bit d'information "1". Les symboles codés à la sortie de l'encodeur sont indiqués sur chacune des branches. Pour le codeur de la figure 2.3, l'arbre correspondant est illustré à la figure 2.5.

- La représentation en treilis

  Le treillis élimine la redondance présente dans l'arbre afin d'obtenir une représentation simplifiée et donc plus facile à utiliser, comme on peut le voir à la figure 2.6. La représentation en treillis permet une bonne compréhension de l'évolution du code et permet de mieux appréhender certains processus de décodage comme ceux utilisés dans la technique de décodage de Viterbi [48] ou encore, comme on le verra par la suite, la technique de décodage des codes Turbo.

2.2.3 Choix d'un bon codeur convolutionnel

Il est important de choisir un bon codeur convolutionnel car d'un codeur à un autre les performances peuvent énormement varier selon qu'on ait choisi

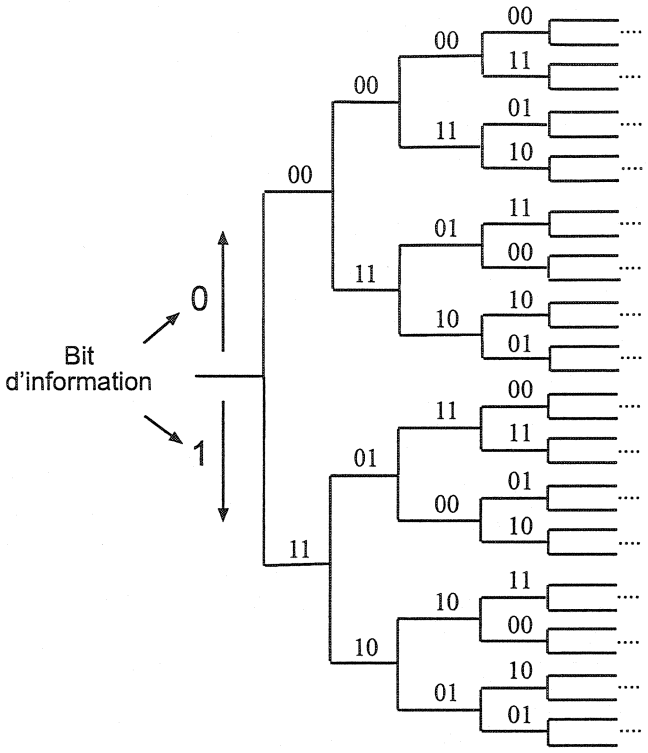

Figure 2.5: Représentation en arbre du codeur de la figure 2.3

un bon ou un mauvais codeur. Les performances d'un code convolutionnel dépendent de ses propriétés de distance [23]. La puissance de correction de ces codes dépend de la longueur de contrainte K, du taux de codage R, ainsi que de certains paramètres de distance tel que la distance libre ou le profil de distance [55].

Soient deux mots de code $\mathbf{X}$ et $\mathbf{Y}$. La distance de Hamming entre les deux mots de code est définie comme le nombre de positions qui différent entre ces deux mots de code. Elle est définie comme suit :

$$d_h(\mathbf{X}, \mathbf{Y}) = W_h(\mathbf{X} \oplus \mathbf{Y}) \tag{2.3}$$

Où W_h est le poids de Hamming. En fait la distance de Hamming est égale au nombre de "1" dans l'opération $(\mathbf{X} \oplus \mathbf{Y})$. La fonction de distance des colonnes notée $d_c(n)$, est la distance de Hamming minimale entre deux mots de code de

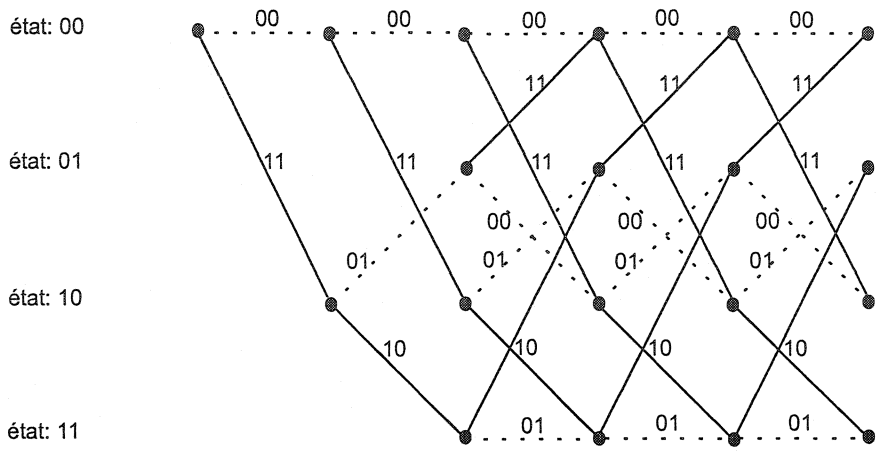

Figure 2.6: Représentation en treillis du codeur de la figure 2.3

longueur n qui différent dans leur première branche.

$$d_c(n) = \ min(d_h(\mathbf{X}_n, \mathbf{Y}_n)) \tag{2.4}$$

Le profil de distance d est constitué de l'ensemble des $d_c(n)$ tel que $n = 1, 2, \ldots, K$. En général il est préférable que le profil de distance croisse le plus rapidement possible.

$$d = (d_c(1), d_c(2), \ldots, d_c(K)) \tag{2.5}$$

La distance minimale d_{min} est définie comme étant la plus petite distance entre le mot de code 0 et tous les autres mots de code de poids non nul. Elle est égale à:

$$d_{min} = d_c(K) \tag{2.6}$$

La distance libre, elle se défini comme :

$$d_{free} = \lim_{n \to \infty} d_c(n) \tag{2.7}$$

La valeur de la distance libre affecte beaucoup plus la capacité de correction du code que la distance minimale. Cette dernière est plutôt plus utile pour un

décodeur qui n'observe que la séquence reçue sur une longueur de contrainte comme le décodeur à seuil.

2.2.4 Codeurs convolutionnels récursifs systématiques

Un codeur convolutionnel est dit systématique si les symboles codés contiennent explicitement les symboles d'information d'entrée. Les codeurs systématiques ne peuvent pas être catastrophiques. Un codeur est catastrophique si un nombre fini d'erreurs sur les symboles codés peut causer un nombre infini de bits d'information décodés en erreur. Cela équivaut au fait qu'on retrouve dans le diagramme d'état une boucle de poids zéro autre que la branche qui boucle sur l'état zéro. Les codeurs systématiques ne peuvent pas être catastrophiques parce qu'on ne peut pas trouver une boucle de poids zéro autre que celle de l'état zéro [33]. Pour de faibles rapports signal sur bruit, ces codeurs offrent légèrement de meilleures performances que les codes non systématiques mais la tendance s'inverse pour des rapports signal sur bruit plus grands [44].

A partir d'un code convolutionnel non systématique on peut construire un code convolutionnel récursif systématique (CRS) en conservant les mêmes vecteurs générateurs. Au niveau du schéma des codeurs récursifs cela signifie que le bit d'information d_k n'est plus directement relié à la première case du registre d'état mais plutôt une information qui est la somme modulo 2 du bit d'information d'entrée et de certaines cases, autre que la première case du registre à décalage. Comme démonstration, prenons l'exemple du codeur de la figure 2.3. A partir de ce code on peut trouver deux codeurs CRS de même taux de codage. $R = 1/2$, de longueur de contrainte $K = 3$ et de séquences génératrices $\mathbf{G}_1 = 1$ et $\mathbf{G}_2 = 7/5$ pour le code de la figure 2.7 et $\mathbf{G}_1 = 1$ et $\mathbf{G}_2 = 5/7$ pour le code de la figure 2.8. La récursivité est donc caractérisée par le fait que les symboles codés sont fonctions des entrées et des sorties précédentes. Au niveau de la matrice génératrice d'un tel codeur cela signifie que la matrice génératrice est une fraction rationnelle irréductible. Le codeur récursif systématique a la même distance libre que le code non systématique à partir duquel il a été construit [21]. Pour un faible rapport signal sur bruit, le code récursif systématique a un léger avantage par rapport au

code convolutionnel non systématique à partir duquel il a été construit [44].

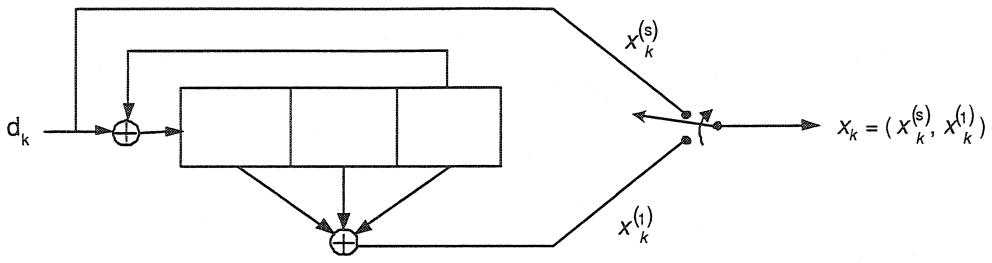

Figure 2.7: Codeur convolutionnel récursif systématique $R = 1/2$, $K = 3$, $\mathbf{G} = [1, 7/5]$

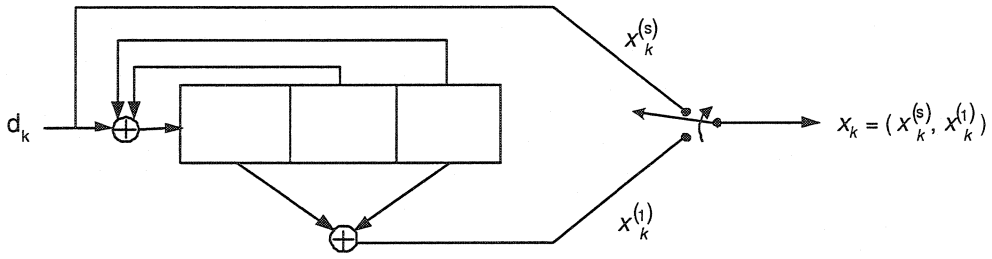

Figure 2.8: Codeur convolutionnel récursif systématique $R = 1/2$, $K = 3$, $\mathbf{G} = [1, 5/7]$

2.2.5 Fonction de transfert des codes convolutionnels

La fonction de transfert représente le rapport qui existe entre la sortie et l'entrée du codeur [22]. A partir de la fonction de transfert on peut déterminer la distance libre du code et la distribution de poids du code. La fonction de transfert se détermine à partir du diagramme d'état. Prenons l'exemple de la figure 2.3, déterminons la fonction de transfert de ce codeur à partir de son diagramme d'état. Ce dernier, est redessiné en divisant l'état zéro en deux, de façon à décrire le fonctionnement exact d'un codeur convolutionnel qui diverge au début de l'état

initial nul puis y reconverge. La fonction de transfert est notée par $T(D, B, L)$ où :

D : Poids de Hamming de la branche.

L : Longueur de la branche.

B : Si la branche correspond à un 1 comme bit d'information.

$T(D, B, L)$ est définie par:

$$T(D, B, L) = \frac{S_f}{S_i} \tag{2.8}$$

S_f et S_i étant respectivement l'état final et l'état initial.

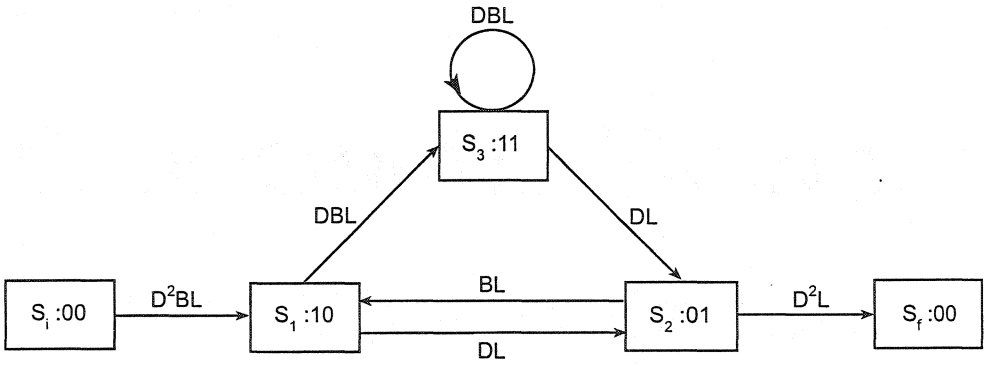

Figure 2.9: Graphe pour le calcul de la fonction de transfert de la figure 2.4

En utilisant la figure 2.9, on peut écrire les équations suivantes:

$$\begin{aligned} S_f &= D^2LS_2 \\ S_2 &= DLS_1 + DLS_3 \\ S_3 &= DBLS_1 + DBLS_3 \\ S_1 &= D^2BLS_i + BLS_2 \end{aligned} \tag{2.9}$$

Des quatre équations précédentes nous pouvons déduire la fonction de transfert qui est :

$$\begin{aligned} T(D,B,L) &= \frac{D^5BL^3}{1-DBL(1+L)} \\ &= D^5BL^3 + D^6L^4(1+L)B^2 + ... + D^{5+l}L^{3+l}(1+L)^lB^{l+1} \\ &= \sum_{k=0}^{\infty} D^{5+k}L^{3+k}(1+L)^kB^{l+k} \\ &= \sum_{k=5}^{\infty} D^kL^{k-2}(1+L)^{k-5}B^{k-4} \end{aligned} \tag{2.10}$$

La forme développée de la fonction de transfert nous permet de déterminer le spectre de distance. La fonction de transfert sera sous la forme d'une somme de termes $D^iL^jB^k$. Cela signifie que nous avons un chemin de poids i, de longueur de branche j et ayant k bits 1 comme entrée. Pour $L = 1$, l'équation 2.10 peut s'écrire:

$$T(D,B) = \sum_{k=5}^{\infty} D^k2^{k-5}B^{k-4} \tag{2.11}$$

L'exposant de la variable D du premier terme de la série de (2.11), correspond à la distance libre du codeur qui égale dans notre exemple à 5.

L'exposant de B donne le nombre de bits d'information égaux à 1. Par conséquent, on peut estimer le nombre de bits en erreur à partir de la fonction de transfert $T(D,B)$ en transmettant une séquence de 0. Ainsi pour trouver le nombre de 1 (bits d'information), on doit dériver la fonction de transfert par rapport à B puis mettre $B = 1$, c'est-à-dire :

$$\begin{aligned} \frac{dT(D,B)}{dB}\Big|_{B=1} &= \sum_{k=0}^{\infty} D^{5+k}2^k(k+1) \\ &= \sum_{k=d_{free}}^{\infty} D^k2^{k-5}(k-4) \\ &= \sum_{k=d_{free}}^{\infty} c_kD^k \end{aligned} \tag{2.12}$$

où c_k est le nombre total des bits en erreur pour tous les chemins ayant un poids de Hamming égal à k.

On peut maintenant évaluer la borne supérieure de la probabilité d'erreur P_B par bit pour un canal gaussien avec modulation BPSK par [22]:

$$P_B \leq \frac{1}{b} \left. \frac{dT(D,B)}{dB} \right|_{B=1,D=P_k} \tag{2.13}$$

$$\leq \frac{1}{b} \sum_{k=d_{free}}^{\infty} c_k P_k \tag{2.14}$$

où b est le numérateur du taux de codage et P_k la probabilité d'erreur pour un chemin incorrect qui diffère du chemin correct entre les k symboles. Elle est définie par:

$$P_k = Q(\sqrt{2kR\frac{E_b}{N_0}}), \quad k \geq d_{free} \tag{2.15}$$

Lorsque le canal est binaire symétrique (BSC) de transition p, P_k est bornée par :

$$P_k < 2^k [p(1-p)]^{1/2} \tag{2.16}$$

En utilisant :

$$Q(\sqrt{x+y}) \leq e^{-\frac{y}{2}} \; Q(\sqrt{x}), \quad x, y \geq 0 \tag{2.17}$$

(2.15) devient :

$$P_k \leq e^{-(k-d_{free})R\frac{E_b}{N_0}} \; Q(\sqrt{2R\frac{E_b}{N_0}d_{free}}), \quad k \geq d_{free} \tag{2.18}$$

Maintenant la probabilité d'erreur P_B peut être bornée par [22]:

$$P_B \leq \frac{1}{b} \; Q(\sqrt{2R\frac{E_b}{N_0}d_{free}}) \; e^{Rd_{free}\frac{E_b}{N_0}} \left. \frac{dT(D,B)}{dB} \right|_{B=1,D=Q(\sqrt{2R\frac{E_b}{N_0}k})} \tag{2.19}$$

2.2.6 Le décodage des codes convolutionnels

La séquence de symboles reçue par le décodeur diffère généralement de la séquence transmise à cause du bruit qui perturbe le canal. Le décodage des codes convolutionnels consiste à retrouver la séquence d'information transmise à partir de la séquence reçue. Cela peut être vu comme l'opération inverse du processus du codage.

Viterbi a développé un algorithme de décodage: l'algorithme à maximum de vraisemblance qui est un décodage probabiliste. Il consiste à déterminer à partir de la séquence reçue le message le plus vraisemblable. Considerons une séquence d'entrée de longueur N, $\mathbf{X}^m$ la séquence codée correspondante et $\mathbf{Y}$, la séquence reçue. $\mathbf{X}^m = (X_i)$, pour $i = 1, 2 \ldots N$ et $\mathbf{Y} = (Y_i)$, pour $i = 1, 2 \ldots N$. Pour minimiser la probabilité d'erreur, il faut choisir le message $\widehat{m}$ tel que :

$$P(\mathbf{Y}|\widehat{m}) \geq P(\mathbf{Y}|m) \qquad \forall \widehat{m} \neq m \tag{2.20}$$

En utilisant la règle de Bayes sur les probabilités conditionnelles dans l'équation 2.20, on peut écrire:

$$P(\widehat{m})P(\mathbf{Y}|\mathbf{X}^{\widehat{m}}) \geq P(m)P(\mathbf{Y}|\mathbf{X}^m) \qquad \forall \widehat{m} \neq m \tag{2.21}$$

où $P(\widehat{m})$ représente la probabilité a priori du message $\widehat{m}$. Si les messages sont équiprobables, l'équation 2.21 devient:

$$P(\mathbf{Y}|\mathbf{X}^{\widehat{m}}) \geq P(\mathbf{Y}|\mathbf{X}^m) \qquad \forall \widehat{m} \neq m \tag{2.22}$$

Minimiser la probabilité d'erreur revient donc à maximiser cette fonction de vraisemblance. Pour un canal discret sans mémoire (DMC) et de probabilité de transition $P(y_i^{(j)}|x_i^{(j)})$, on peut écrire pour la branche i:

$$P(\mathbf{Y}_i|\mathbf{X}_i^{\widehat{m}}) = \prod_{j=1}^{V} P(y_i^{(j)}|x_i^{(j)}) \tag{2.23}$$

Afin d'utiliser une mesure additive et plus pratique, on utilise le logarithme de $P(\mathbf{Y}_i|\mathbf{X}_i^{\widehat{m}})$. Le résultat représente alors la mesure de vraisemblance pour la branche i,

$$\gamma_i = \sum_{j=1}^{V} log(P(y_i^{(j)}|x_i^{(j)})) \tag{2.24}$$

Pour une séquence de N bits d'informations, la fonction de vraisemblance est:

$$\Gamma = \sum_{i=1}^{N} \gamma_i \tag{2.25}$$

où Γ, γ_i et $log(P(y_i^{(j)}|x_i^{(j)}))$ sont appelés respectivement métrique cumulative totale, métrique de branche et métrique de symbole. Cette dernière est habituellement arrondie à un entier.

Le processus de décodage basé sur l'observation de $\mathbf{Y}$ consiste à choisir le mot de code dont la métrique cumulative est maximale. Pour le canal binaire symétrique, le mot de code ayant la métrique cumulative maximale est le chemin dont la distance de Hamming est minimale [22]. Mais la difficulté du décodage provient plutôt du fait que le nombre de chemins à considérer est très grand pour permettre une recherche exhautive. L'algorithme de Viterbi est basé sur l'addition des métriques de branche, pour chaque noeud, le chemin retenu sera celui dont la métrique cumulative est la plus grande. Au cas où deux chemins reconvergent en un même état avec la même métrique cumulative, on choisit l'un des deux au hasard. L'avantage de cette technique est que l'effort de calcul pour un nombre de branche reste constant, contrairement à la plupart des autres techniques de décodage. Par contre la taille du treillis augmente exponentiellement avec la longueur de contrainte du code.

2.3 La modulation BPSK

Il existe de nombreux types de modulation pour les communications numériques. Néanmoins la principale modulation que nous allons utiliser dans ce mémoire, celle

utilisée dans les codes turbo est la modulation BPSK. Par ailleurs elle est également la plus utilisée dans les sytèmes de communications numériques. L'expression d'un signal BPSK est la suivante:

$$\begin{aligned} s_0(t) &= -(\sqrt{2\frac{E_b}{T_b}})\cos 2\pi f_p t \qquad 0 \leq t \leq T_b \\ s_1(t) &= +(\sqrt{2\frac{E_b}{T_b}})\cos 2\pi f_p t \qquad 0 \leq t \leq T_b \end{aligned} \tag{2.26}$$

où f_p est la fréquence de l'onde sinusoïdale utilisée pour la transmission d'un symbole codé, T_b la durée d'un symbole et E_b l'énergie transmis par bit.

2.4 Canal de transmission

Dans un système de communication, un canal physique s'occupe de l'acheminement des données. Il existe plusieurs types de canaux tels que les câbles (fils torsadés, cable coaxial, fibre optique), des guides d'onde, l'atmosphère. Ce canal est un véritable problème pour la transmission de données car il est une source de bruit modifiant ainsi l'information transmise. Il existe essentiellement deux modèles pour représenter les canaux de transmission : le canal gaussien et le canal de Rayleigh.

2.4.1 Canal gaussien

Le canal à bruit additif blanc gaussien : ("Additif White Gaussien Noise" en anglais (AWGN)) est utilisé pour modéliser les canaux satellites. Le bruit est un processus aléatoire, $n(t)$, stationnaire gaussien de moyenne nulle et de variance σ^2. Soit $s(t)$ le signal transmis, $n(t)$ le bruit qui s'ajoute au signal et $r(t)$ le signal reçu. Le signal reçu s'exprime par la relation:

$$r(t) = s(t) + n(t) \tag{2.27}$$

La densité de probabilité $p(n)$ de ce bruit est:

$$p(n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{n^2}{2\sigma^2}} \tag{2.28}$$

2.4.2 Canal de Rayleigh

Il est utilisé pour représenter les canaux terrestres et radiomobiles. Dans ces canaux, les signaux reçus présentent des évanouissements dûs aux chemins multiples (réflexions provoqués par les obstacles physiques). A cela s'ajoute l'effet Doppler dû au déplacement du mobile ce qui se traduit par une variation de la fréquence proportionnellement à la vitesse du mobile. Le modèle gaussien ne suffit plus pour modéliser ce canal, le signal reçu s'exprime plutôt sous la forme:

$$r(t) = \alpha s(t) + n(t) \tag{2.29}$$

La densité de probabilité de la variable α, s'exprime sous la forme:

$$p(\alpha) = \begin{cases} \frac{\alpha}{\sigma^2} e^{-\frac{\alpha^2}{2\sigma^2}} & \text{si} \quad \alpha \geq 0 \\ 0 & \text{autrement} \end{cases} \tag{2.30}$$

2.5 Conclusion

Dans ce chapitre nous avons présenté les éléments de base d'un système de communication numérique. Nous avons donné des informations sur la modélisation des canaux et la modulation BPSK. Nous avons également présenté plus en détail la structure des codes convolutionnels et un algorithme de décodage de ces derniers. Nous avons ainsi présenté les codes convolutionnels qui sont un élément important des codes turbo. Ces derniers seront présentés dans le chapitre suivant.

CHAPITRE 3

LES CODES TURBO

3.1 Introduction

Parmi les techniques de codage actuelles, les codes turbo introduits par Berrou, Glavieux et Thitimajshima en 1993 [7] sont actuellement les plus puissants, en terme de probabilité d'erreur. Cette dernière se rapproche de très près (0.7 dB) de la limite théorique de Shannon pour une probabilité d'erreur de 10^{-5}. Le codeur turbo est constitué de deux codeurs convolutionnels concaténés en parallèle et séparés par un entrelaceur. Le décodeur turbo quant à lui est constitué de deux décodeurs spécifiques à chaque codeur convolutionnel, ces décodeurs sont concaténés en série et séparés par un entrelaceur. Le codeur turbo tient son nom de sa technique de décodage itératif et du fait que d'un décodeur à un autre on a un échange d'information.

Chaque décodeur se sert non seulement du bit de parité du codeur concerné mais également de l'information qui a été transmise par l'autre décodeur et ce processus est repété plusieurs fois jusqu'à la saturation des décodeurs. Dans ce chapitre nous détaillons les différents éléments qui interviennent dans le codeur turbo à savoir les entrelaceurs et l'algorithme de décodage.

3.2 Présentation du codeur turbo

La notion de concaténation a été introduite pour la première fois par Forney [21]. Il existe deux sortes de concaténation, la concaténation série et la concaténation parallèle. Le terme série indique que la sortie d'un codeur constitue l'entrée de l'autre codeur séparé ou non par un entrelaceur. Le taux de codage global du

codeur série R_g est égal à : $R_1 R_2$, R_1 et R_2 étant les taux de codage des codeurs. Au niveau de la concaténation parallèle, les codeurs reçoivent en entrée les mêmes bits d'informations. Il faut noter que, que ce soit la concaténation parallèle ou sérielle, le nombre de codeurs peut être plus de deux. Le codeur turbo tel qu'introduit par Berrou [7] est constitué de deux codeurs convolutionnels identiques concaténés en parallèle et séparés par un entrelaceur. Les bits d'informations sont reçus à l'entrée des deux décodeurs mais dans un ordre différent pour le deuxième codeur. Le taux de codage global R_g du codeur turbo est égale à :

$$\begin{aligned} R_g &= \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} - 1} \\ &= \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2} \end{aligned} \tag{3.1}$$

Le principe du codeur turbo est présenté à la figure 3.1.

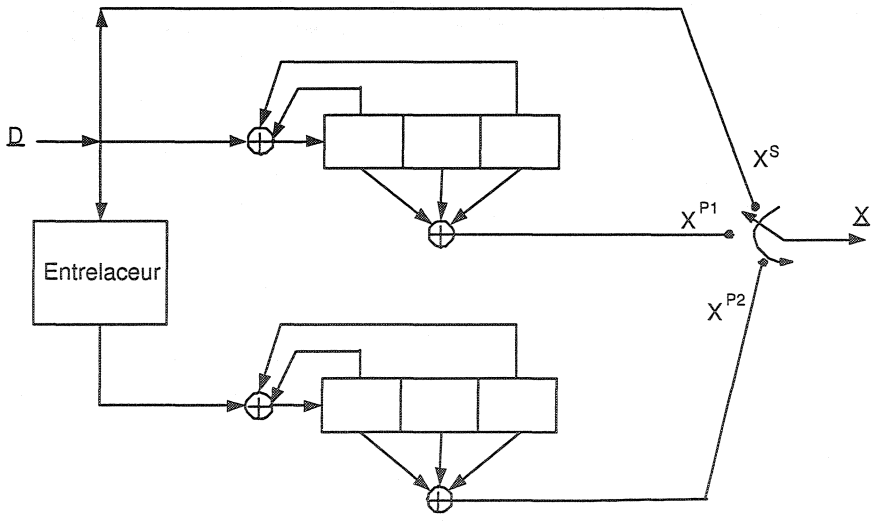

Figure 3.1: Schéma de principe d'un codeur turbo

3.3 Les entrelaceurs

L'entrelaceur prend en entrée des symboles dans un ordre donné et les sort dans un ordre temporel différent. Il permet de créer des codes avec de meilleures propriétés de distance. On pourra utiliser des codes avec de petites longueurs de contrainte comme s'il s'agissait de codes avec de grandes longueurs de contrainte [2], ce qui simplifierait l'algorithme de décodage.

L'entrelaceur permet en outre de répartir les erreurs dans un bloc d'information mais son rôle principal est de décoreller les symboles d'entrée des deux décodeurs. La condition pour que le principe de décodage itératif puisse s'appliquer est que les entrées des décodeurs soient indépendantes. Dans ce cas le LLR (Log Likehood Ratio) pourra se décomposer en trois éléments : l'information systématique, l'information a priori et l'information extrinsèque. Il est donc important de décoreller les entrées de ces décodeurs. Si les séquences d'entrée sont décorellées, il y a une forte probabilité que la plupart des erreurs non corrigées par le premier décodeur le soient par le deuxième. Les erreurs non détectées par le premier seront réparties par l'entrelaceur, le deuxième décodeur pourra les corriger plus facilement.

L'entrelaceur de par sa fonction influence fortement la capacité de correction du codeur, d'autant plus que la puissance des codes turbo provient essentiellement de son décodage itératif. Il est donc important de choisir un bon entrelaceur, il existe dans la littérature plusieurs types d'entrelaceurs. L'entrelaceur sera choisi en fonction de la taille du bloc d'information, de la complexité et de la performance recherchée. Les entrelaceurs sont classés en trois grandes catégories : les entrelaceurs blocs, hélicoïdaux, les entrelaceurs convolutionnels et les entrelaceurs aléatoires. [27] et [50] donnent de plus amples renseignements sur les entrelaceurs. Le désentrelaceur quand à lui réalise l'opération inverse de l'entrelaceur, c'est à dire qu'il remet les symboles dans leur ordre d'origine.

3.3.1 Les entrelaceurs blocs

Les entrelaceurs blocs sont caractérisés par un traitement sur un bloc de données, en inscrivant les symboles d'information dans une matrice $m * n$, m étant le nombre de lignes et n le nombre de colonnes. La lecture se fait selon plusieurs procédés dépendant de l'entrelaceur bloc utilisé. Nous présentons dans la suite les différents entrelaceurs blocs qui existent dans la littérature [50].

3.3.1.1 L' entrelaceur bloc classique

Au niveau de l'entrelaceur bloc classique, les symboles d'informations sont écrits ligne par ligne dans une matrice $n * m$ et lus colonne par colonne. Il existe plusieurs possibilités de lecture : la lecture peut se faire de gauche à droite ou de droite à gauche pour les colonnes et du haut vers le bas ou du bas vers le haut pour les lignes. Ce qui nous donne quatre possibilités de lecture, elles sont illustrées à la figure 3.2, un exemple de lecture est également donné à la figure 3.3.

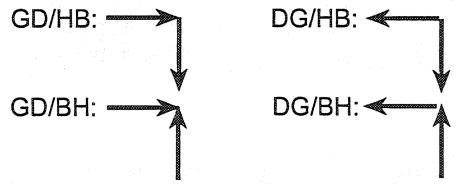

Figure 3.2: Modes de lecture de l'entrelaceur bloc classique

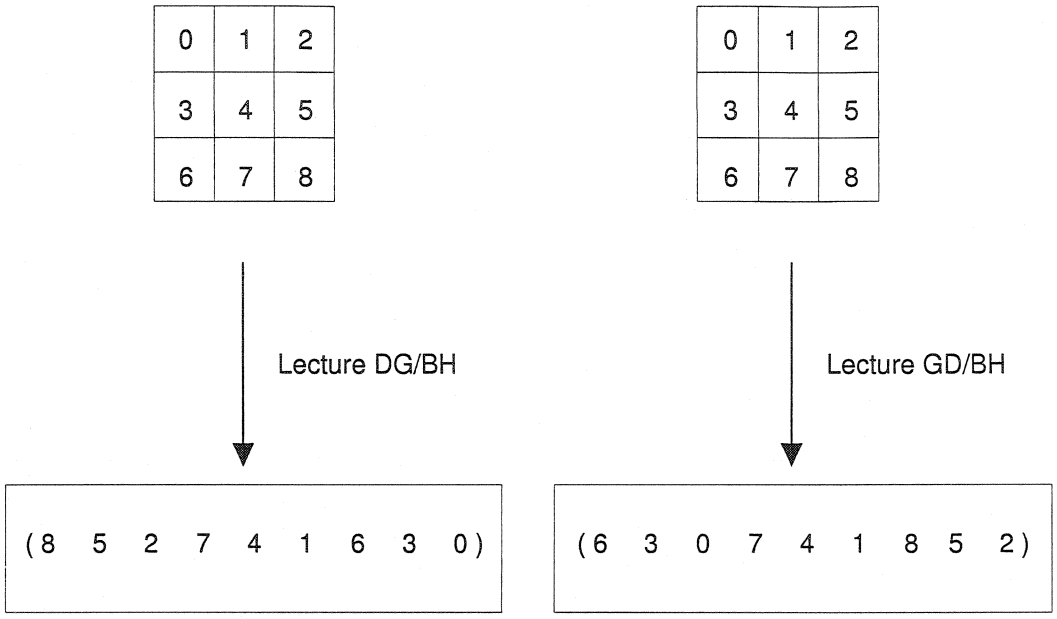

Figure 3.3: Entrelaceur bloc classique

3.3.2 Les entrelaceurs hélicoïdaux

Au niveau des entrelaceurs hélicoidaux l'écriture se fait colonne par colonne et non plus ligne par ligne comme dans le cas des entrelaceurs blocs classiques. Il existe plusieurs types d'entrelaceurs hélicoidaux:

3.3.2.1 Les entrelaceurs blocs hélïcoidaux

Ces entrelaceurs ont été introduits par Chi [12], [13]. Dans ce type d'entrelaceur la lecture s'effectue par diagonale, et elle peut se faire de plusieurs manières selon qu'il s'agisse du type I, II ou III. Les entrelaceurs de type III sont caractérisés par :

Processus d'écriture:

$$m = k\%M \tag{3.2}$$

$$n = [k \ / \ M] \tag{3.3}$$

Processus de lecture:

$$m = k\%M \tag{3.4}$$

$$n = (k\%M + [k/G] + [k/M]R)\%N \tag{3.5}$$

avec:

$$R = (M-1)\%N \tag{3.6}$$

$$G = MN/(N,R) \tag{3.7}$$

L'opérateur % est l'opérateur modulo.
M étant le nombre de lignes, N le nombre de colonnes, n l'indice de la colonne, m l'indice de la ligne, k le $k^{\text{ième}}$ symbole de la séquence d'information, (N,R) le plus grand commun diviseur de N et R et $[X]$ la partie entière de X.

Dans le cas où $R = 0$, on se retrouve avec l'entrelaceur de type I, le processus de lecture devient $n = k\%N$, un exemple de l'entrelaceur de type I est montré à la figure 3.4.

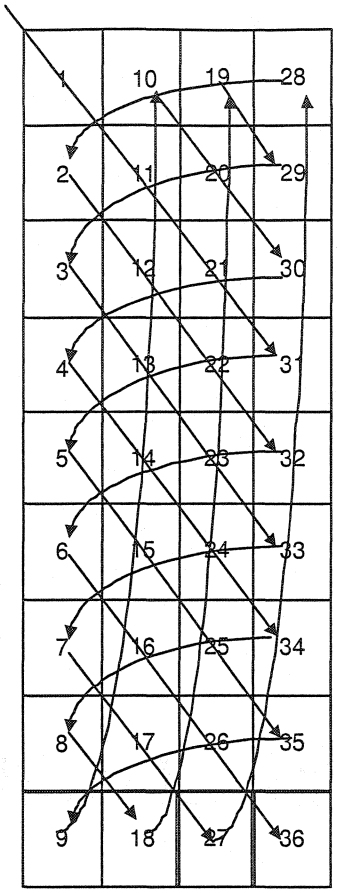

Figure 3.4: Entrelaceur de type I

Après lecture de la matrice, la séquence entrelacée est telle que représentée à la figure 3.5.

1	11	21	31	5	15	25	35	9	10	20	30	4	14	24	34	8	18
19	29	3	13	23	33	7	17	27	28	2	12	22	32	6	16	26	36

Figure 3.5: Séquence de sortie de l'entrelaceur type I

L'entrelaceur de type II correspond au cas où N et R sont premiers entre eux. La différence avec le type I est que lorsque qu'on atteint la dernière ligne on ne change

pas de colonne contrairement au type I. Ce type d'entrelaceur est illustré à la figure 3.6. Après lecture de la matrice, la séquence entrelacée est telle que représentée à la figure 3.7. Il faut également signaler qu'il existe d'autres possibilités de lecture en diagonale de la matrice. On peut effectuer la lecture en commencant par le bas, ce qui donnerait de meilleurs résultats que le type I [31]. Un exemple est illustré dans la figure 3.8. A une probabilité d'erreur de $10^{-}5$, pour N égal à 900, 4 itérations et un code de mémoire 4, Lajnef a effectué des simulations, l'entrelaceur de la figure 3.8 est de 0.5 dB meilleur à celui de la figure 3.7.

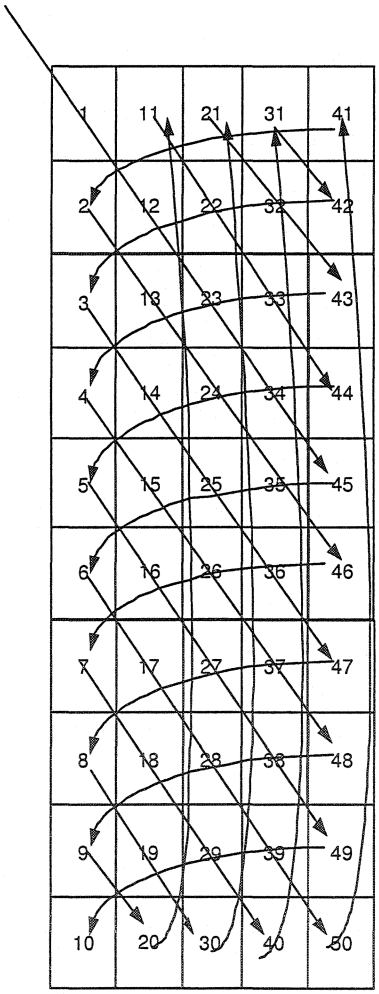

Figure 3.6: Entrelaceur de type II

1	12	23	34	45	6	17	28	39	50
41	2	13	24	35	46	7	18	29	40
31	42	3	14	25	36	47	8	19	30
21	32	43	4	15	26	37	48	9	20
11	22	33	44	5	16	27	38	49	10

Figure 3.7: Séquence de sortie de l'entrelaceur type II

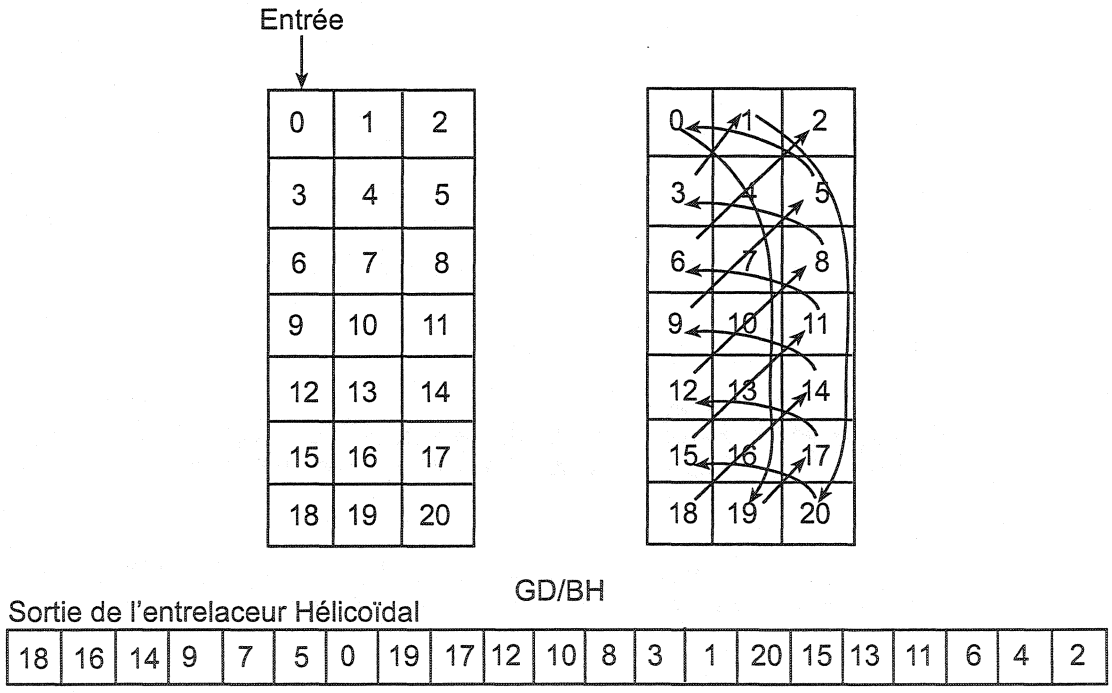

Figure 3.8: Entrelaceur GD/BH

3.3.2.2 L'entrelaceur d'ordre N

Il a été introduit par Chi [13], [12]. Il est de type hélicoïdal parce que les bits sont inscrits hélicoïdalement dans une matrice. L'ordre N signifie qu'il existe $N-1$ colonnes. Dans ce type d'entrelaceur on remplit N lignes des symboles d'entrée et ensuite on passe à la colonne suivante, mais en passant à la colonne

suivante on commence le remplissage en décalant d'une ligne par rapport à la colonne précédente. Une caractéristique de cet entrelaceur est que les bits qui se suivaient avant entrelacement seront espacés d'au moins $N-1$ après l'entrelacement. Dans la figure 3.9 qui suit nous montrons un exemple d'un entrelaceur d'ordre 4 pour 24 symboles d'informations.

1		
2	5	
3	6	9
4	7	10
13	8	11
14	17	12
15	18	21
16	19	22
	20	23
		24

Figure 3.9: Entrelaceur d'ordre N=4

Après lecture de la matrice, la séquence entrelacée est telle que représentée à la figure 3.10.

1	*	*	2	5	*	3	6	9	4	7	10	13	8	11
14	17	12	15	18	21	16	19	22	*	20	23	*	*	24

Figure 3.10: Séquence de sortie de l'entrelaceur d'ordre N=4

3.3.2.3 L'entrelaceur d'ordre (M,N)

Il a été introduit par Chi [13], [12]. Dans ce type d'entrelaceur on a une matrice de m lignes et de n colonnes. Le processus est un peu différent de l'entrelaceur d'ordre N mais le principe reste le même. La condition pour ce type d'entrelaceur est que M et N soient des nombres premiers entre eux [36]. Le $k^{\text{ième}}$ mot de code est inscrit dans la colonne $((k-1).M) modulo\ N$ mais en commencant à la ligne k. Les autres symboles restant sont insérés en complétant les lignes du début qui avaient été laissées. La séquence entrelacée est trouvée en lisant la matrice ligne par ligne. La figure 3.11 qui suit nous montre un exemple d'entrelaceur d'ordre BH(8,5).

1	23	37	16	30
2	24	38	9	31
3	17	39	10	32
4	18	40	11	25
5	19	33	12	26
6	20	34	13	27
7	21	35	14	28
8	22	36	15	29

Figure 3.11: Entrelaceur hélicoïdal (8,5)

Après lecture de la matrice, la séquence entrelacée est telle que représentée à la figure 3.12.

1	23	37	16	30	2	24	38	9	31	3	17	39	10	32	4	18	40	11	25
5	19	33	12	26	6	20	34	13	27	7	21	35	14	28	8	22	36	15	29

Figure 3.12: Séquence de sortie de l'entrelaceur hélicoïdal (8,5)

3.3.3 Les entrelaceurs convolutionnels

3.3.3.1 L' entrelaceur convolutionnel

Les entrelaceurs convolutionnels sont caractérisés par deux facteurs, le délai D et le nombre de lignes m. Ils ont été introduits par Ramsey [37]. Le schéma de principe et sa matrice correspondante pour un codeur convolutionnel sont montrés

dans les figures : 3.13 et 3.14. Un commutateur réparti les bits d'informations sur les différentes lignes du codeur convolutionnel. Le premier bit est introduit sur la première ligne, le deuxième sur la deuxième ligne et ainsi de suite. Le $(k+1)$ bit sera introduit sur la première ligne et le processus recommence. Il faut souligner que chaque $i^{\text{ième}}$ ligne introduit un retard de iD. Pour avoir la séquence de sortie un autre commutateur balaie les différentes lignes. Les délais introduits par les lignes font qu'un bit de la $i^{\text{ième}}$ ligne sera lu pour la première fois au $i^{\text{ième}}$ passage du commutateur. La séquence entrelacée est trouvée en lisant la matrice colonne par colonne.

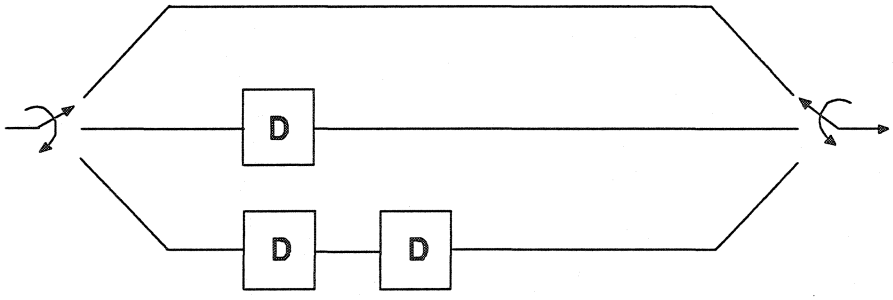

Figure 3.13: Entrelaceur convolutionnel de 3 lignes

1	4	7	10	13	···		
*	2	5	8	11	14	···	
*	*	3	6	9	12	15	···

Figure 3.14: Matrice de l'entrelaceur convolutionnel de la figure 3.13, D=1

3.3.3.2 L'entrelaceur à décalage cyclique

Comme son nom l'indique, dans ce type d'entrelaceur il s'agit de décaler les lignes m, ou les lignes et les colonnes n de la matrice de l'entrelaceur avec la condition $m \leq n$. Les bits d'information sont insérés dans la matrice colonne par

colonne. Il y a deux types d'entrelaceur : l'entrelaceur à décalage cyclique simple et celui à décalage cyclique double. Pour l'entrelaceur à décalage cyclique simple on décale uniquement les lignes. La $i^{\text{ième}}$ ligne est décalée de $i * (D-1)$ vers la gauche, D étant la partie entière de $\frac{n}{m}$. Dans le cas du décalage cyclique double, on décale les lignes et les colonnes. Les lignes se décalent comme dans le cas de l'entrelaceur à décalage cyclique simple. Les colonnes quant à elles peuvent être décalées de plusieurs manières. La meilleure serait de décaler chaque colonne j, de $j * D \ mod(m)$, D étant un entier $\leq \frac{n}{m}$. Le décalage de la colonne se fera vers le bas ou vers le haut si respectivement j est pair ou impair. Un exemple de l'entrelaceur double cyclique est montré à la figure 3.15. Dans les 2 types d'entrelaceurs la lecture se fait colonne par colonne.

$$\begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 \end{bmatrix} \begin{array}{c} \text{double décalage} \\ \longrightarrow \\ \text{cyclique} \end{array} \begin{bmatrix} 15 & 18 & 7 & 17 & 20 & 16 & 12 \\ 1 & 4 & 14 & 3 & 6 & 2 & 19 \\ 8 & 11 & 21 & 10 & 13 & 9 & 5 \end{bmatrix}$$

Figure 3.15: Entrelaceur à double décalage cyclique, $m = 3$, $n = 7$ et $D = 2$

3.3.4 Les entrelaceurs aléatoires

Comme son nom l'indique, dans l'entrelaceur aléatoire [50], les bits de sortie sont choisis de manière aléatoire cependant on s'assure qu'un bit d'entrée ne soit choisi qu'une seule fois. Etant donné qu'on ne peut pas savoir a priori où est-ce qu'un bit va se retrouver, il est nécessaire de garder une table de correspondance afin de pouvoir délacer les bits d'informations. Il existe plusieurs variantes au niveau des entrelaceurs aléatoires : l'entrelaceur pair-impair, symétrique et S-aléatoire.

3.3.4.1 L'entrelaceur pair-impair

Dans l'entrelaceur pair-impair, les bits sont choisis aléatoirement mais avec la contrainte qu'un bit conserve sa parité après entrelacement.

3.3.4.2 L'entrelaceur symétrique

Au niveau de cet entrelaceur le principe est l'échange de positions entre 2 bits d'informations. Si le bit 1 et le bit 2 occupaient respectivement les positions i et j, après entrelacement ces bits occuperont respectivement les positions j et i. Ce type d'entrelaceur a pour avantage d'utiliser moins d'espace mémoire que l'entrelaceur aléatoire, la moitié, ce qui peut être assez intéressant pour des entrelaceurs de grande taille. Un exemple de cet entrelaceur est décrit à la figure 3.16.

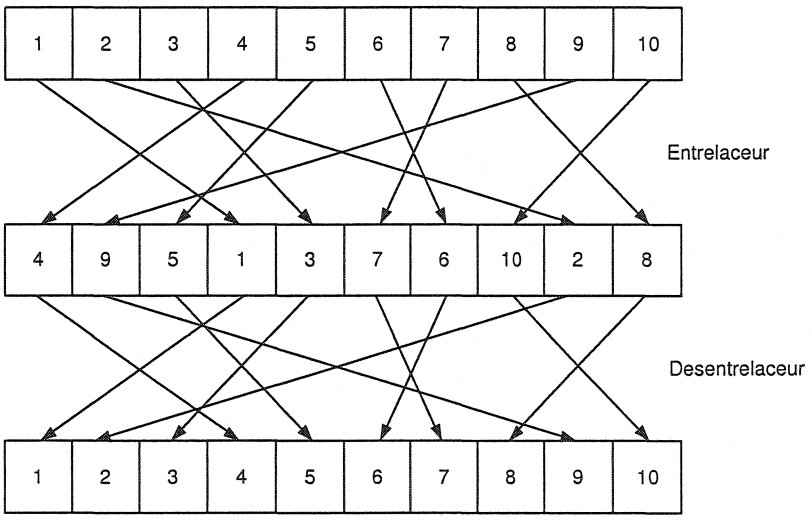

Figure 3.16: Entrelaceur et désentrelaceur symétrique

3.3.4.3 Les entrelaceurs S-aléatoires

Le principe de cet entrelaceur est d'entrelacer les symboles d'informations de telle sorte que 2 symboles d'informations successifs soient espacés, après entrelacement d'une distance d'au moins S. Ces entrelaceurs ont été proposés par Divsalar et Pollar [17]. Les meilleurs résultats sont obtenus pour $S = \sqrt{\frac{N}{4}}$. Cet algorithme peut s'avérer assez complexe surtout quand S est grand. La figure suivante 3.17 illustre ce type d'entrelaceur.

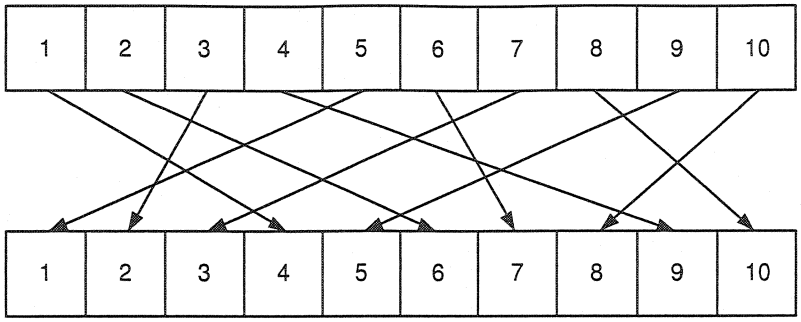

Figure 3.17: Entrelaceur S aléatoire, S=2

3.4 La terminaison des codeurs

A chaque transmission d'un bloc de données les codeurs doivent être réinitialisés. La réinitialisation du codeur turbo n'est pas aussi simple que celui du codeur convolutionnel à cause de la récursivité des codeurs et de la présence de l'entrelaceur. Ici les deux codeurs n'ont pas le même état final. Il existe dans la littérature plusieurs méthodes de réinitialisation des codes turbo. Divsalar D. et Pollara [16] proposent de reboucler le codeur sur lui même. Dans ce cas le codeur ne doit pas tenir compte des (K-1) bits utilisés pour la réinitialisation. Reed et Pietrobon quand à eux proposent de modifier l'initialisation de la métrique arrière (voir la section 3.5.1.2). Dans ce cas, on n'aurait pas besoin de transmettre de queue [39]. La métrique d'état en arrière est initialisée comme suit pour l'algorithme Log-MAP:

$$B_N^i(m) = -(K-1)\ ln2 \quad \text{pour tout m} \neq 0 \tag{3.8}$$

Les notions de métriques avant et arrière seront présentées plus en détail dans la section 3.5.1.

3.5 Le décodage turbo

La puissance de correction des codes turbo est en partie due à sa technique de décodage. La technique utilisée est un décodage itératif, le décodage s'effectuant

en plusieurs itérations. Le décodeur global est constitué de deux décodeurs, chaque décodeur étant associé à un codeur convolutionnel. Les décodeurs fournissent à leurs sorties une valeur du logarithme de rapport de vraisemblance en anglais, (Log Likehood Ratio, LLR) du bit d'information d_k à décoder. Le LLR peut se décomposer en trois éléments : l'information extrinsèque Le_k, l'information a priori La_k et l'information systématique ou intrinsèque Li_k. L'information extrinsèque est transmise au décodeur suivant et devient son information a priori. Le décodeur se sert non seulement du bit de parité et du bit systématique lui correspondant mais également d'une information provenant de l'autre décodeur, l'information a priori. Le décodage s'effectue ainsi jusqu'à saturation des décodeurs. Le principe de décodage est décrit à la figure 3.18.

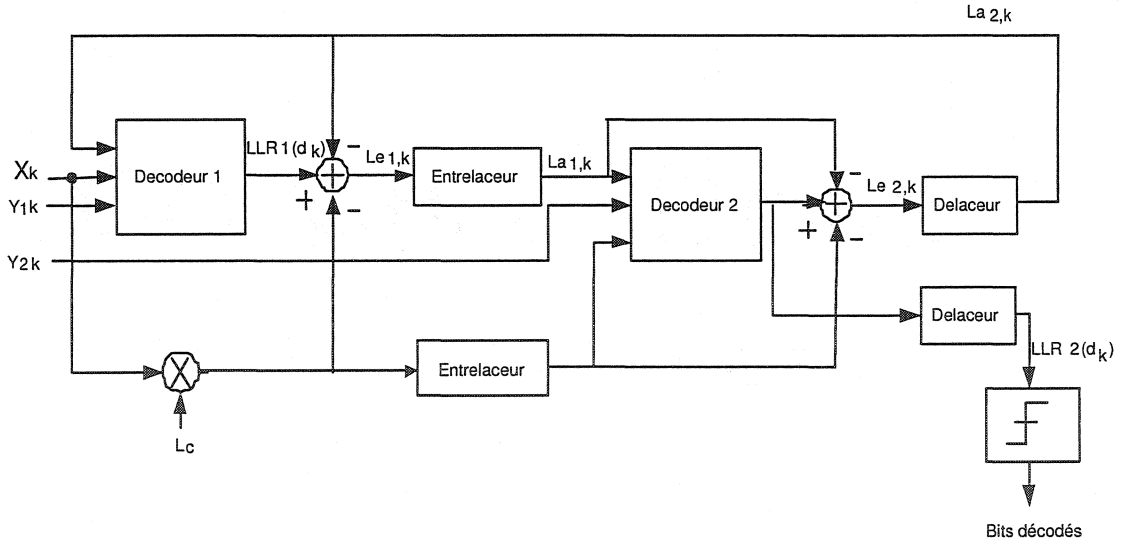

Figure 3.18: Principe du décodeur itératif turbo

Y_{1k}, Y_{2k} sont respectivement les bits de parité des codeurs convolutionnels 1 et 2 constituant le codeur turbo et $L_c = 2/\sigma^2$, σ^2 est la variance du bruit affectant le canal. Nous verrons dans la suite que l'expression du LLR peut se

décomposer en trois éléments que sont l'information systématique, l'information a priori et l'information extrinsèque (section 3.5.1.3). L'information systématique est constitué du bit systématique affecté du coefficient L_c, d'où la présence de ce coefficient après le bit systématique au niveau du schéma du décodeur turbo.

Il existe dans la littérature plusieurs algorithmes de décodage : l'algorithme à maximum a postériori (MAP), le LogMAP, le MaxLogMAP, le soft output Viterbi algorithm (SOVA) [24]. Ce dernier est une variante de l'algorithme de Viterbi et est relativement moins complexe que l'algorithme MAP, mais les simplifications entraînent une perte d'optimalité allant jusqu'à 1 dB [40]. Par ailleurs dans l'algorithme de Viterbi, la probabilité d'erreur par symbole n'est pas minimisée mais plutôt la probabilité d'erreur par séquence. Tandis que dans le cas de l'algorithme MAP, c'est le contraire. L'algorithme MAP a été développé par Bahl, Cocke [1] et adapté par Berrou, Glavieux et Thitimajshima [7]. L'algorithme MAP est assez complexe, il demande de nombreux calculs. Des simplifications ont été apportées à cet algorithme: l'algorithme LogMAP a été développé, il présente les mêmes performances que le MAP et a l'avantage d'utiliser moins de calculs. C'est celui que nous utilisons pour nos simulations. Une autre simplification de cet algorithme a été introduit: l'algorithme MaxLogMAP, on obtient cependant une baisse de performances de 0.75dB [40].

3.5.1 L'algorithme MAP (Maximum A Postériori

Cet algorithme a été inventé par Bahl, Cocke et adapté par Berrou, Glavieux et Thitimajshima pour le décodage turbo. C'est un algorithme optimal c'est à dire qu'il minimise la probabilité d'erreur par bit. C'est un algorithme à maximum a posteriori c'est à dire qu'à partir d'une séquence reçue on choisit le bit d_k dont la probabilité d'envoi est la plus élévée. Pour ce faire il faut déterminer le rapport de vraisemblance, le LLR :

$$LLR(d_k) = ln\frac{P(d_k = 1|\mathbf{R}_1^N)}{P(d_k = 0|\mathbf{R}_1^N)} \qquad k = 1, 2, \ldots, n, \ldots, N. \tag{3.9}$$

avec $\mathbf{R}_1^N$ la séquence reçue de longueur N, d_k et $\hat{d}_k$, respectivement les bits reçu et décodé. La règle de décision se définit comme suit:

$$\begin{aligned} LLR(d_k) &\geq 0 : \text{le bit décodé est } \widehat{d_k} = 1 \\ LLR(d_k) &< 0 : \text{le bit décodé est } \widehat{d_k} = 0 \end{aligned} \tag{3.10}$$

Développons l'expression du LLR afin de pouvoir le simplifier et retrouver les éléments qui interviennent dans le décodage itératif, à savoir l'information systématique, l'information extrinsèque et l'information a priori. En introduisant la probabilité conjointe, λ_k^i

$$\lambda_k^i = P(d_k = i, S_k = m|\mathbf{R}_1^N) \quad i = 0, 1. \tag{3.11}$$

La probabilité a posteriori du bit d_k devient donc:

$$P(d_k = i, |\mathbf{R}_1^N) = \sum_{m=0}^{M-1} \lambda_k^i(m), \tag{3.12}$$

Le LLR peut alors se définir par:

$$LLR(d_k) = ln \frac{\sum_0^{M-1} \lambda_k^1(m)}{\sum_0^{M-1} \lambda_k^0(m)} \tag{3.13}$$

S_k étant l'état du codeur à l'instant k et $M = 2^{K-1}$ la mémoire du codeur. Définissons $\alpha_k^i(m)$, $\beta_k^i(m)$ et $\gamma_i(R_k, m', m)$ comme étant respectivement la métrique d'état en avant, la métrique d'état en arrière et la métrique de branche.

$$\alpha_k^i(m) = P(d_k = i, S_k = m, \mathbf{R}_1^k) \tag{3.14}$$

$$\beta_k^i(m) = P(\mathbf{R}_{k+1}^N | d_k = i, S_k = m) \tag{3.15}$$

$$\gamma_i(\mathbf{R}_k, m', m) = P(d_k = i, S_k = m, \mathbf{R}_k | d_{k-1} = j, S_{k-1} = m') \tag{3.16}$$

De (3.11), la probabilité conjointe peut être réécrite:

$$\lambda_k^i(m) = \frac{P(d_k = i, S_k = m, \mathbf{R}_1^k, \mathbf{R}_{k+1}^N)}{P(\mathbf{R}_1^N)} \tag{3.17}$$

Nous obtenons ainsi:

$$\lambda_k^i(m) = \frac{P(d_k = i, S_k = m, \mathbf{R}_1^k) P(\mathbf{R}_{k+1}^N | d_k = i, S_k = m, \mathbf{R}_1^k)}{P(\mathbf{R}_1^N)} \tag{3.18}$$

En tenant compte du fait que les évènements après l'instant k ne sont pas influencés par l'observation $\mathbf{R}_1^k$ si le bit d_k et l'état du codeur S_k sont connus, (3.18) devient :

$$\lambda_k^i(m) = \frac{P(d_k = i, S_k = m, \mathbf{R}_1^k) P(\mathbf{R}_{k+1}^N | d_k = i, S_k = m)}{P(\mathbf{R}_1^N)} \tag{3.19}$$

La conjointe devient donc:

$$\lambda_k^i(m) = \frac{\alpha_k^i(m) \beta_k^i(m)}{P(\mathbf{R}_1^N)} \tag{3.20}$$

Finalement de (3.13) et (3.20), le LLR peut s'exprimer sous la forme:

$$LLR(d_k) = ln \frac{\sum_0^{M-1} \alpha_k^1(m) \beta_k^1(m)}{\sum_0^{M-1} \alpha_k^0(m) \beta_k^0(m)} \tag{3.21}$$

3.5.1.1 Développement des métriques

Il s'agit de développer la métrique d'état en avant et la métrique d'état en arrière sous forme récursive et de simplifier la métrique d'état. Le développement et la simplification de ces métriques ont été faits dans l'annexe I. Il en ressort que:

$$\alpha_k^i(m) = \sum_{m'=0}^{M-1} \sum_{j=0}^{1} \alpha_{k-1}^j(m') \gamma_i(\mathbf{R}_k, m', m) \tag{3.22}$$

$$\beta_k^i(m) = \sum_{m'=0}^{M-1} \sum_{j=0}^{1} \beta_{k+1}^j(m') \gamma_j(\mathbf{R}_{k+1}, m, m') \tag{3.23}$$

3.5.1.2 Différentes étapes de l'algorithme MAP

- Etape 1 Les probabilités $\alpha_0^i(m)$ et $\beta_N^i(m)$ sont initialisées comme suit:

$$\alpha_0^i(0) = 1 \quad \alpha_0^i(m) = 0 \quad \forall m \neq 0, \; i = 0, 1 \tag{3.24}$$

$$\beta_N^i(0) = 1 \quad \beta_N^i(m) = 0 \quad \forall m \neq 0, \; i = 0, 1 \tag{3.25}$$

- Etape 2 Pour chaque observation $\mathbf{R}_k$, les probabilités $\alpha_k^i(m)$ et $\gamma_i(\mathbf{R}_k, m', m)$ sont calculées en utilisant respectivement (3.22) et (I.16).

- Etape 3 Quand la séquence $\mathbf{R}_1^N$ a été complètement reçue, les probabilités $\beta_k^i(m)$ sont calculées en utilisant la relation (3.23), ensuite on multiplie les probabilités $\alpha_k^i(m)$ et $\beta_k^i(m)$ pour obtenir $\lambda_k^i(m)$. Finalement le LLR associé à chaque bit d_k est calculé à partir de la relation (3.13).

3.5.1.3 L'information extrinsèque

Nous avons vu à la section 3.5 que le LLR du bit à décoder d_k peut se décomposer en trois élements à savoir l'information systématique ou intrinsèque Li_k, l'information extrinsèque Le_k et l'information a priori La_k. C'est uniquement l'information extrinsèque qui est transmise au décodeur suivant et elle devient son information a priori. Dans la sous-section précédente nous avons calculé le LLR. Nous allons maintenant le décomposer en ses trois éléments: l'information systématique, extrinsèque et a priori. L'équation (3.9) nous donne l'expression du LLR :

$$
\begin{aligned}
LLR(d_k) &= ln\left[\frac{P(d_k=1|\mathbf{R}_1^N)}{P(d_k=0|\mathbf{R}_1^N)}\right] \\
&= ln\left[\frac{P(\mathbf{R}_1^N, d_k=1)}{P(\mathbf{R}_1^N, d_k=0)}\right] \\
&= ln\left[\frac{P(\mathbf{R}_1^{k-1}, \mathbf{R}_{k+1}^N, x_k, y_k|d_k=1)\times P(d_k=1)}{P(\mathbf{R}_1^{k-1}, \mathbf{R}_{k+1}^N, x_k, y_k|d_k=0)\times P(d_k=0)}\right] \\
&= ln\left[\frac{P(\mathbf{R}_1^{k-1}, \mathbf{R}_{k+1}^N, y_k|d_k=1)}{P(\mathbf{R}_1^{k-1}, \mathbf{R}_{k+1}^N, y_k|d_k=0)}\right] + ln\left[\frac{P(x_k|d_k=1)}{P(x_k|d_k=0)}\right] + ln\left[\frac{P(d_k=1)}{P(d_k=0)}\right]
\end{aligned}
\tag{3.26}
$$

Le décodeur reçoit comme bit systématique $x_k = 2d_k - 1 + n_k$, n_k étant le bruit affectant le canal et d_k le bit systématique transmis à l'instant k. Dans le cas d'un canal gaussien, la variable x_k est une variable gaussienne de moyenne $(2d_k - 1)$ et dans ce cas nous avons :

$$
P(x_k|d_k=i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x_k-(2i-1))^2} \tag{3.27}
$$

On déduit de cette dernière équation que (3.26) peut s'écrire sous la forme:

$$
\begin{aligned}
LLR(d_k) &= \frac{2}{\sigma^2}x_k + Le_k + La_k \\
&= Li_k + Le_k + La_k
\end{aligned}
\tag{3.28}
$$

Au cas où il s'agit d'un canal de Rayleigh, la variable x_k sera une variable aléatoire de moyenne $m_\alpha(2d_k - 1)$ et dans ce cas (3.26) devient:

$$
LLR(d_k) = m_\alpha \frac{2}{\sigma^2}x_k + Le_k + La_k \tag{3.29}
$$

où

$$\begin{cases} Le_k &= ln\left[\frac{P(\mathbf{R}_1^{k-1},\mathbf{R}_{k+1}^{N},y_k|d_k=1)}{P(\mathbf{R}_1^{k-1},\mathbf{R}_{k+1}^{N},y_k|d_k=0)}\right] \\ m_\alpha \frac{2}{\sigma^2} x_k &= ln[\frac{P(x_k|d_k=1)}{P(x_k|d_k=0)}] \\ La_k &= ln[\frac{P(d_k=1)}{P(d_k=0)}] \end{cases}$$

3.5.2 L'algorithme LogMAP

L'algorithme MAP est très lourd à implémenter car il néccessite beaucoup de calcul, notamment beaucoup d'exponentielles. L'algorithme LogMAP a été mis en oeuvre pour palier à ce problème. Il s'agit du même algorithme sauf qu'on le développe dans le domaine logarithmique, ce qui permet d'éliminer les opérations exponentielles et donc de diminuer considérablement la complexité de l'algorithme. Le LogMAP présente pratiquement les mêmes performances que l'algorithme MAP. Il fait introduire un nouvel opérateur : $\mathbf{E}$ [19].

$$\begin{aligned} e^{x_1} + e^{x_2} &= e^{x_1}(1 + e^{x_2 - x_1}) \\ &= e^{x_2}(1 + e^{x_1 - x_2}) \\ &= e^{\max(x_1,x_2) + \ln(1 + e^{-|x_1 - x_2|})} \end{aligned} \tag{3.30}$$

De (3.30) nous avons :

$$\begin{aligned} \ln(e^{x_1} + e^{x_2}) &= \max(x_1, x_2) + \ln(1 + e^{-|x_1 - x_2|}) \\ &= \max(x_1, x_2) + f_c(|x_1 - x_2|) \end{aligned} \tag{3.31}$$

où la fonction $f_c(.)$ est une fonction de correction qui peut être trouvée en utilisant une table. L'expression $\ln(e^{x_1} + e^{x_2} + \ldots + e^{x_n})$ est appelée l'opérateur $\mathbf{E}$ et peut être calculée de façon récursive comme suit :

$$\begin{aligned}
\mathbf{E}_n(x(n)) &= ln(\sum_n e^{x(n)}) \\
&= \ln(e^{x_1} + e^{x_2} + \ldots + e^{x_n}) \\
&= \ln(\triangle + e^{x_n}), \qquad \triangle = e^{x_1} + e^{x_2} + \ldots + e^{x_{n-1}} = e^x \\
&= \max(\ln \triangle, x_n) + \ln(1 + e^{-|\ln \triangle - x_n|}) \\
&= \max(x, x_n) + \ln(1 + e^{-|x - x_n|}) \qquad (3.32)
\end{aligned}$$

On définit de nouvelles métriques en avant, en arrière et d'état comme suit :

$$\underline{\alpha}_k^i(m) = ln(\alpha_k^i(m)) \qquad (3.33)$$

$$\underline{\beta}_k^i(m) = ln(\beta_k^i(m)) \qquad (3.34)$$

$$\underline{\gamma}_i(\mathbf{R}_k, m', m) = ln(\gamma_i(\mathbf{R}_k, m', m)) \qquad (3.35)$$

L'expression (3.21) du LLR peut se réecrire comme suit :

$$\begin{aligned}
LLR(d_k) &= ln\frac{\sum_0^{M-1} \alpha_k^1(m)\beta_k^1(m)}{\sum_0^{M-1} \alpha_k^0(m)\beta_k^0(m)} \\
&= ln\frac{\sum_0^{M-1} e^{\underline{\alpha}_k^1(m) + \underline{\beta}_k^1(m)}}{\sum_0^{M-1} e^{\underline{\alpha}_k^0(m) + \underline{\beta}_k^0(m)}} \\
LLR(d_k) &= \mathbf{E}_{m=0}^{M-1}\left(\underline{\alpha}_k^1(m) + \underline{\beta}_k^1(m)\right) - \mathbf{E}_{m=0}^{M-1}\left(\underline{\alpha}_k^0(m) + \underline{\beta}_k^0(m)\right) \qquad (3.36)
\end{aligned}$$

De (3.33), (3.35) et (3.22) nous avons une nouvelle expression de $\underline{\alpha}_k^i(m)$:

$$\begin{aligned}
\underline{\alpha}_k^i(m) &= ln(\alpha_k^i(m)) \\
&= ln(\sum_{m'=0}^{M-1} \sum_{j=0}^{1} \gamma_i(\mathbf{R}_k, m', m)\alpha_{k-1}^j(m')) \\
&= ln \sum_{m'=0}^{M-1} \sum_{j=0}^{1} e^{\underline{\gamma}_i(\mathbf{R}_k, m', m) + \underline{\alpha}_{k-1}^j(m')} \qquad (3.37)
\end{aligned}$$

Cette dernière expression peut-être calculée en utilisant (3.31).

De même de (3.34) et (3.23) nous avons :

$$\begin{aligned}\underline{\beta}_k^i(m) &= ln(\beta_k^i(m)) \\ &= ln(\sum_{m'=0}^{M-1}\sum_{j=0}^{1}\gamma_j(\mathbf{R}_{k+1},m,m')\beta_{k+1}^j(m')) \\ &= ln\sum_{m'=0}^{M-1}\sum_{j=0}^{1}e^{\underline{\gamma}_j(\mathbf{R}_{k+1},m,m')+\underline{\beta}_{k+1}^j(m')} \end{aligned} \tag{3.38}$$

Cette dernière expression peut également être calculée en utilisant (3.31).

L'initialisation des métriques se trouve modifiée et se présente comme suit :

$$\underline{\alpha}_0^i(0) = 0 \quad \underline{\alpha}_0^i(m) = -\infty \quad \forall m \neq 0,\ i = 0,1 \tag{3.39}$$

$$\underline{\beta}_N^i(0) = 0 \quad \underline{\beta}_N^i(m) = -\infty \quad \forall m \neq 0,\ i = 0,1 \tag{3.40}$$

3.5.3 L'algorithme MaxLogMAP

L'algorithme MaxLogMAP est une simplification de l'algorithme LogMAP dans lequel (3.32) se réduit uniquement à l'opération max donnant:

$$\begin{aligned}\mathbf{E}_n(x(n)) &= ln(\sum_n e^{x(n)}) \\ &= \ln\left(e^{x_1}+e^{x_2}+\ldots+e^{x_n}\right) \\ &= \ln\left(\triangle+e^{x_n}\right), \qquad \triangle = e^{x_1}+e^{x_2}+\ldots+e^{x_{n-1}} = e^x \\ &= \max\left(\ln\triangle, x_n\right) \\ &= \max\left(x, x_n\right) \end{aligned} \tag{3.41}$$

Cette simplification se traduit par une perte de performance d'environ 0.75 dB [40].

Le tableau 3.19 fait une comparaison de la complexité des algorithmes MAP,

LogMAP, MaxLogMAP et SOVA [50].

	MAP	LogMAP	MaxLogMAP	SOVA
Addition	$2.2_k.2_\nu + 6$	$6.2_k.2_\nu + 6$	$4.2_k.2_\nu + 8$	$2.2_k.2_\nu + 9$
multiplication	$5.2_k.2_\nu + 8$	$2_k.2_\nu$	$2.2_k.2_\nu$	$2_k.2_\nu$
op max		$4.2_\nu - 2$	$4.2_\nu - 2$	$2.2_\nu - 1$
exponentiel	$2.2_k.2_\nu$			

Figure 3.19: Table comparative de la complexité de calcul des différents algorithmes

3.6 Performances d'erreur des codes turbo

Les performances des codes turbo peuvent varier énormément selon plusieurs facteurs : la longueur de contrainte, le type de l'entrelaceur, le nombre d'itérations, les vecteurs générateurs des codes convolutionnels composant le code turbo. Tous ces facteurs influencent énormément les performances d'un code turbo. [31] traite plus en détail l'influence de tous ces facteurs sur les performances des codes turbo.

Au niveau des entrelaceurs pour une longueur d'entrelaceur $N \leq 196$, le meilleur entrelaceur se trouve être l'entrelaceur symétrique avec S=7 mais étant donné sa complexité, il serait meilleur d'utiliser un entrelaceur bloc. Pour des longueurs d'entrelaceur supérieures à 196, l'entrelaceur symétrique s'avère toujours être le meilleur mais il demande un grand effort de calcul. Le meilleur choix serait l'entrelaceur aléatoire qui donne nettement de meilleures performances par rapport à l'entrelaceur bloc, hélicoidal et à l'entrelaceur convolutionnel. Pour une probabilité d'erreur de 10^{-5}, on a une différence de 0.6 dB. [42] donne de plus amples détails sur l'influence de l'entrelaceur sur les performances des codes turbo.

Le nombre d'itérations influence également grandement les performances d'un code turbo [7]. Par exemple, de l'itération 2 à l'itération 18 et pour une probabilité d'erreur de 10^{-5} on a une différence de 2.4 dB. Il faut noter cependant que la

complexité du décodage augmente linéairement en fonction du nombre d'itérations. Cependant à partir d'une certaine itération les décodeurs tendent à saturer. Il faut donc définir des critères d'arrêts qui permettront de stopper le décodage. Il existe plusieurs critères d'arrêts dans la littérature : l'entropie croisée(EC) [25], le sign-change-ratio(SCR) [43], le critère Hard Decision Aided(HDA) [43] et le HDA mixed.

- **Entropie croisée**

  Ce critère est basé sur la similitude entre deux distributions, il a été présenté dans [25], on arrête le décodage quand l'expression

$$T_2(i) \approx \sum_{k=1}^{N} \frac{\triangle\,(Le_k^2(i))^2}{e^{|LLR_k^1(i)|}} \text{ avec } \triangle\, Le_k^2(i) = Le_k^2(i) - Le_k^2(i-1)\ , i \geq 1 \quad (3.42)$$

  est approximativement comprise entre 10^{-2} et 10^{-4}. Ce critère peut être appliqué également au niveau du décodeur 1 mais cela accroît la complexité du décodage.

- **Sign-change-ratio**

  Ce critère est basé sur le changement de signe $C(i)$ de l'information extrinsèque de l'itération $i-1$ à l'itération i. On décide généralement d'arrêter le décodage lorsque le rapport entre $C(i)$ et N, la longueur de la séquence d'information est inférieur à $0.03 \sim 0.005$. Ce critère a été défini par Shao, Lin et Fossorier [43].

- **Hard decision aided**

  Le HDA est basé sur le changement de signe du LLR de l'itération i et du LLR de l'iteration $(i-1)$. S'il ont le même signe on arrête le décodage. Ce critère a été défini dans [43]. Il existe aussi un autre critère semblable à celui-ci (le HDA mixed) mais qui consiste à étudier le changement de signe du LLR du décodeur 1 à l'itération i et celui du décodeur 2 à l'itération $(i-1)$.

3.7 Conclusion

Dans ce chapitre nous avons décrit tous les éléments qui interviennent dans les codes turbo. Nous avons présenté son décodage itératif, des différents entrelaceurs qui existent. Nous avons décrit les différents algorithmes de décodage utilisés. Les codes turbo utilisent la modulation BPSK, cependant pour des applications à largeur de bande réduite, cette modulation n'est pas très intéressante. Le chapitre suivant fera donc l'objet de l'étude des schémas de modulation élévées associées aux codes turbo.

CHAPITRE 4

ETUDE DES MODULATIONS MULTI-NIVEAUX: SCHÉMA TURBO PRAGMATIQUE

4.1 Introduction

Le développement des communications numériques est tel que les systèmes de communication doivent répondre à des exigences de plus en plus poussées, notamment en matière de débit de transmission et de fiabilité de l'information et ceci malgré les limitations en largeur de bande. L'efficacité spectrale η est définit comme le nombre de bits d'information transmis à chaque instant T, elle se traduit mathématiquement par l'expression:

$$\eta = R\ log_2 M \tag{4.1}$$

R, étant le taux de codage et M le dégré de la modulation. Cette relation met en évidence le dilemme suivant: l'augmentation de l'efficacité spectrale passe par une augmentation des paramètres R et M, et donc par une diminution de la qualité de la transmission si l'on considère que le rapport signal sur bruit ne change pas. On rappelle que la probabilité d'erreur augmente en fonction du taux de codage et de l'ordre de la modulation.

Les codes turbo présentés dans le chapitre précédent permettent d'atteindre des performances d'erreurs remarquables, rapprochant la limite théorique de Shannon de 0.7 dB. Malheureusement la modulation BPSK utilisée par ces codes n'est pas intéressante pour les applications à largeur de bande limitée. Il serait donc intéressant d'associer les codes turbo à des schémas de modulations élévées, [8] et

[41]. Il en existe plusieurs. Nous présenterons d'abord ceux qui existent dans la littérature et ensuite nous étudierons les plus intéressants, à savoir le turbo pragmatique et la modulation codée en treillis. Ils feront respectivement l'objet de ce chapitre et du suivant.

4.2 Le codage multi-niveaux

Ce type de codage a été présenté par Imai et Hirakawa [29] en 1977. C'est une méthode qui combine le codage et la modulation et qui utilise plusieurs codes correcteurs d'erreurs binaires, le schéma de principe est décrit à la figure 4.1. Etant donné que les codes turbo donnent des performances d'erreurs remarquables, nous utiliserons comme codes correcteurs d'erreurs, des codeurs turbo. Pour une modulation M-aire, on utilise $m = log_2 M$ codeurs turbo comme décrit à la figure 4.2. Soit D une séquence d'information de longueur N. Cette séquence est divisée en m blocs, $D = (D_1, D_2, \ldots, D_m)$. Chaque message bloc D_i est codé individuellement par le codeur turbo i. Si k_i est la longueur du bloc d'information D_i et n_i la longueur de la séquence codée, chaque codeur i aura un taux de codage $R_i = \frac{k_i}{n_i}$. On obtient:

$$R = \frac{\sum_{i=1}^{m} k_i}{\sum_{i=1}^{m} n_i} = \frac{k}{mn} \tag{4.2}$$

Pour plus de simplicité, nous supposerons que toutes les séquences codées ont la même longueur n. Dans ce cas le taux de codage global R du codeur multi-niveaux est égale à : $\frac{k}{nm}$. L'efficacité spectrale du codeur est donnée par :

$$\eta = R\ log_2\ M = \frac{k}{n} \quad \text{bits/s/Hz} \tag{4.3}$$

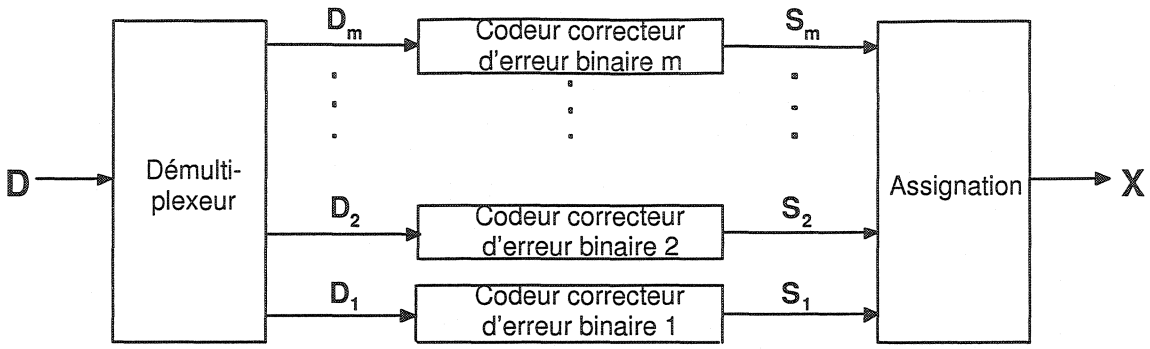

Figure 4.1: Codeur multi-niveaux

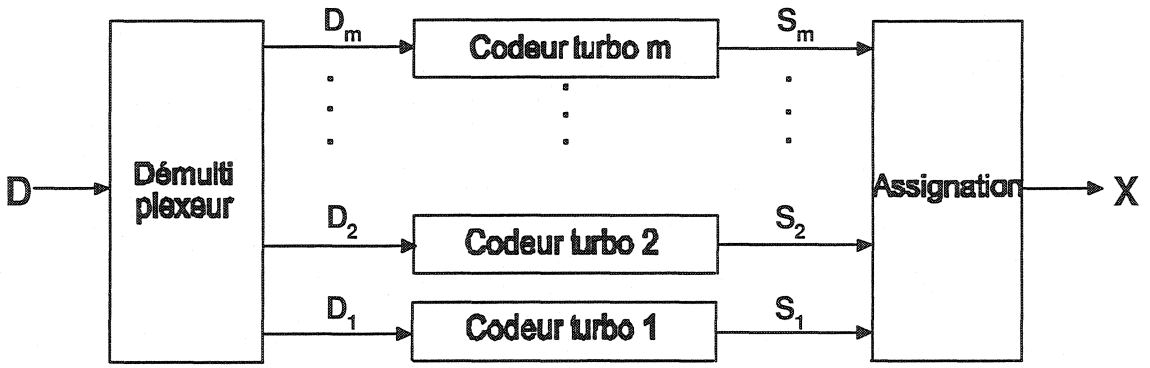

Figure 4.2: Codeur turbo multi-niveaux

Le décodeur à maximum de vraisemblance opère sur le treillis et est en général trop complexe à implémenter. Alternativement une technique de décodage sous optimale appelée décodage multi-étages peut être utilisée. Le décodeur multi-étages emploie m décodeurs. Chaque décodeur étant adapté au code correcteur d'erreur binaire lui correspondant. Lorsque nous utilisons le cas particulier du codeur multi-niveaux de la figure 4.2, les décodeurs utilisés sont des décodeurs turbo binaires. Chaque codeur i est décodé successivement par le décodeur i correspondant. Les décisions du décodeur seront transmises aux décodeurs de plus haut niveau. Le décodeur i ne travaillera pas seulement en fonction de la séquence reçue mais également des décisions $\hat{x^1}, \hat{x^2}, \ldots, \hat{x^{i-1}}$ prises par les

décodeurs précédents. Le schéma de principe d'un décodeur multi-étages est présenté à la figure 4.3. Le cas particulier du décodeur turbo multi-niveaux est présenté à la figure 4.4.

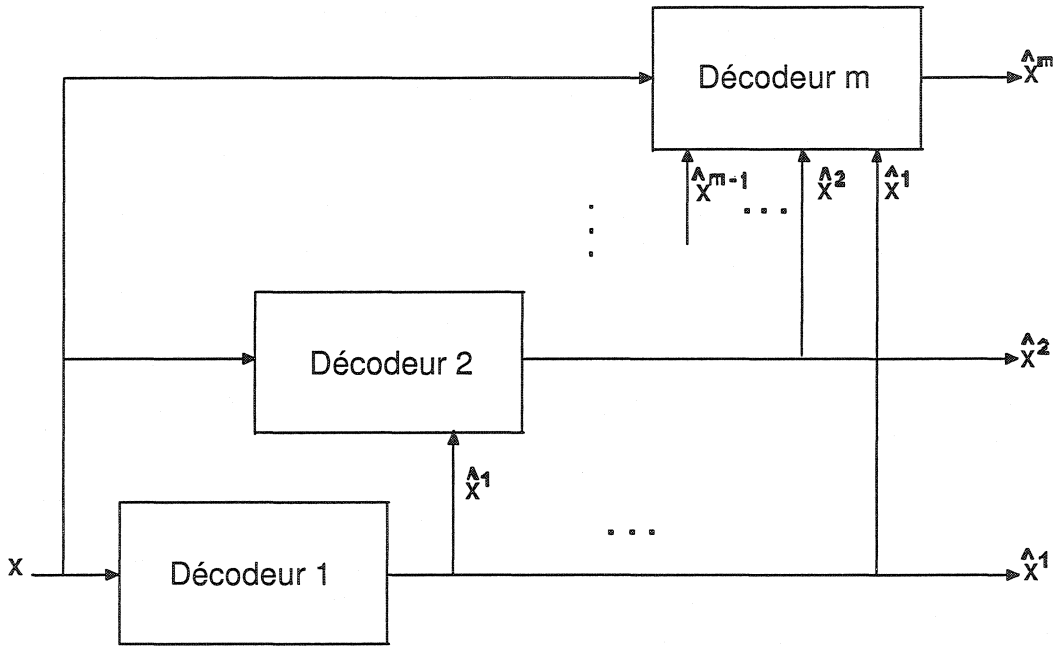

Figure 4.3: Décodeur multi-niveaux

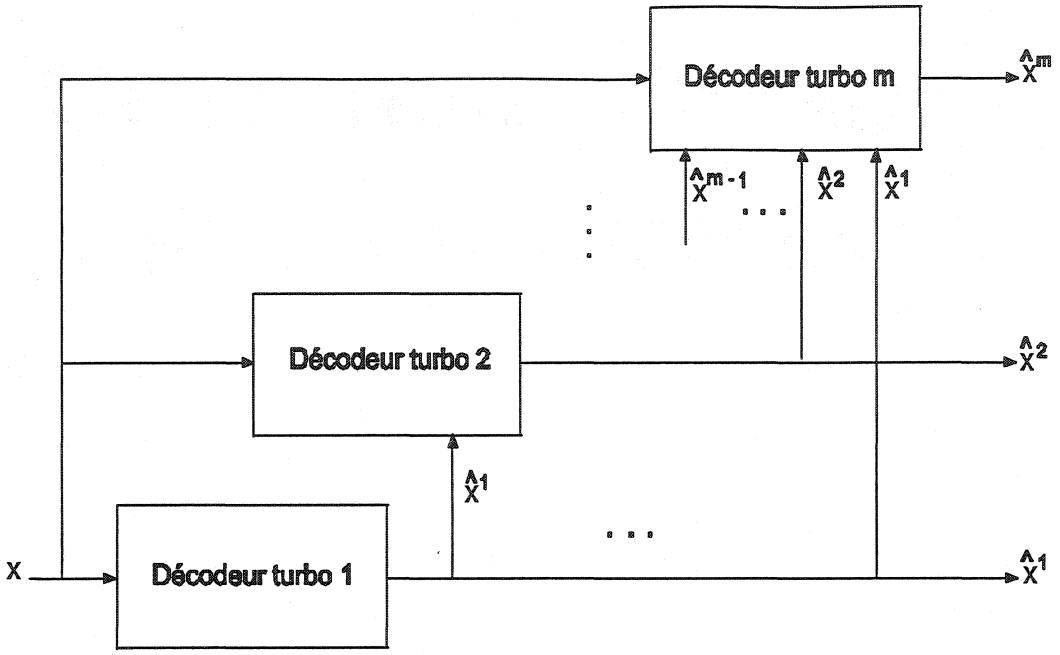

Figure 4.4: Décodeur turbo multi-niveaux

Un autre élément important est de choisir convenablement le taux de codage des codeurs. Waschmann et Huber [51], [52] ont proposé une méthode pour choisir ces taux. Dans cette méthode le taux de codage, à un niveau i donné est choisi égal à la capacité du canal binaire équivalent à ce niveau. Etant donné que la capacité totale du canal est égale à la somme de toutes les m capacités, on trouve ainsi les différents taux de codage. Pour des codes de longueur infinie, on arrive à effectuer une transmission sans erreur. Les performances des codes turbo permettent d'avoir des propagations d'erreurs négligeables d'un étage à un autre. C'est un facteur important qui nous permet d'utiliser un décodage multi-niveaux. Néanmoins pour des longueurs de blocs plus petits, il peut avoir une perte de performance significative [51].

Pour un ordre M de modulation on utilise $m = log_2 M$ codeurs turbo pour le codeur multi-niveaux et le même nombre pour son décodeur. Nous savons que la présence d'un entrelaceur entraîne déjà un délai important de longueur N, sans compter la complexité due à la longueur de contrainte. Utiliser m codeurs turbo

revient à multiplier l'effort de calcul et le délai de codage par m. Au niveau du décodage les mêmes inconvénients se présentent puisque le décodeur multi-niveaux utilise aussi m décodeurs turbo binaires. La lourdeur du décodage itératif ne fera que se multiplier par$m$. Malgré l'énorme complexité du schéma multi-niveaux ses performances ne sont pour autant pas meilleures à celles des autres schémas [35]. Nous n'etudierons, par conséquent, plus le schéma multi-niveaux.

4.3 Schéma Turbo Trellis Coded Modulation (TTCM)

Le schéma de modulation multi-niveaux TTCM a été présenté par Robertson [41]. Il consiste en la concaténation turbo de deux codeurs TCM (Trellis Coded Modulation), codes développés pour la première fois par Ungerboeck [45]. Les TCM combinent la modulation et le codage en optimisant la distance euclidienne entre les mots de code. Ils peuvent être décodés par l'algorithme de Viterbi ou de Bahl-Jelinek [1]. L'idée de Robertson était d'utiliser les avantages en bande passante du TCM et la puissance de décodage itératif des codes turbo. La figure 4.5 nous présente la structure générale d'un codeur TTCM.

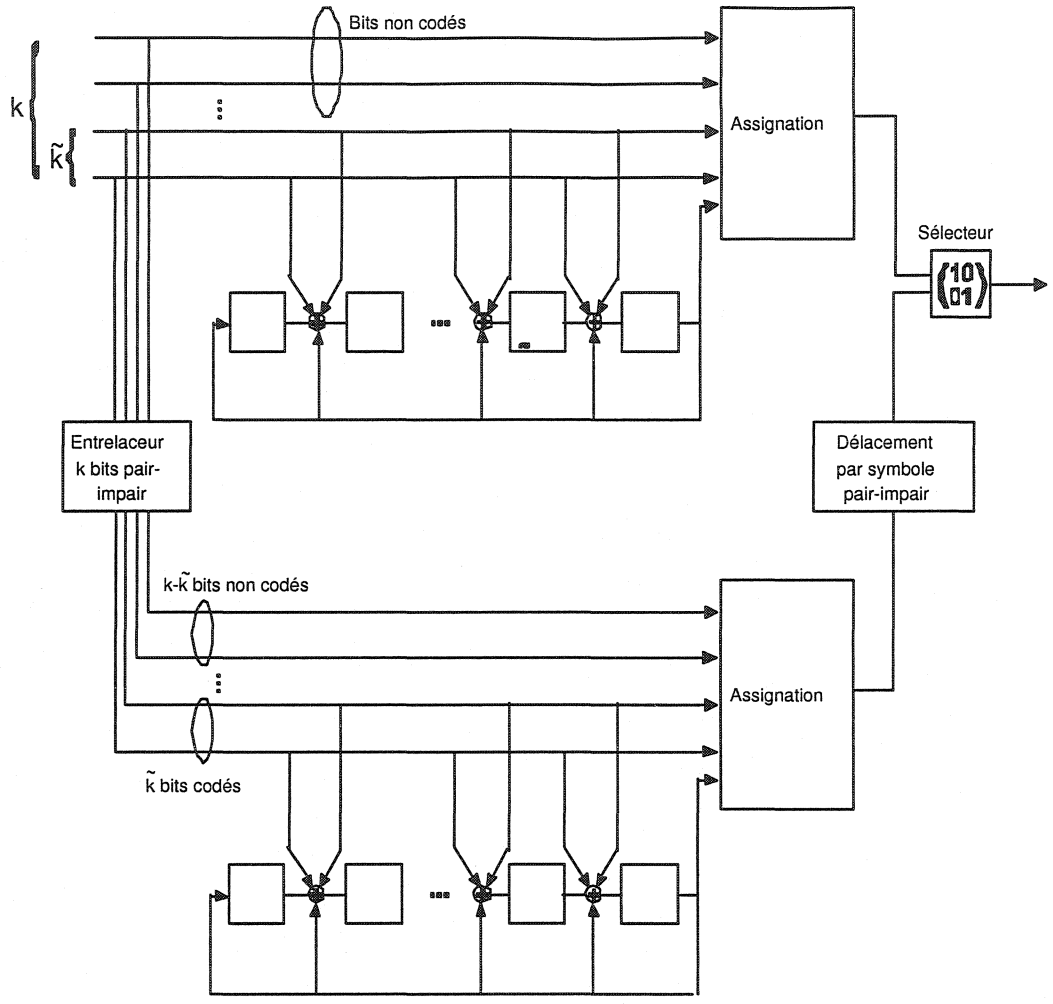

Figure 4.5: Structure générale d'un codeur TTCM

Le taux de codage R des codeurs TCM est de $k/k+1$, avec $k = m-1$, m étant le nombre de bits utilisés pour former un symbole. Les codeurs TCM sont reliés par un entrelaceur de symboles qui opère non sur des bits mais plutôt sur des groupes de k bits. L'opération de l'entrelaceur s'effectue sur des groupes de k bits ce qui signifie que k bits sont déplacés à la fois et non un seul bit. L'entrelaceur subdivise les bits d'information par groupes de k bits et effectue l'entrelacement par groupes. Si au niveau de l'entrelaceur de bits classique le bit i occupe la position j après entrelacement, au niveau de l'entrelaceur k bits, les bits i, $i+1$, ..., $i+k$ occuperont respectivement les positions j, $j+1$, ..., $j+k$. Le schéma TTCM exige

que l'entrelaceur soit de type pair-impair, dont les détails seront donnés dans le chapitre suivant. Une fois les bits codés, ils sont assignés à une constellation qui peut être soit de type M-PSK ou M-QAM avec $k+1 = log_2 M$. Le partitionnement utilisé est une partition selon Ungerboeck qui optimise la distance euclidienne entre les signaux. La sortie du deuxième codeur est désentrelacée par un désentrelaceur de symbole. Cette fonction permet d'assurer que les k bits ayant servi à déterminer les $k+1$ bits codés du premier et du deuxième codeur soient les mêmes à un instant donné. Les mêmes bits d'information se retrouvent donc dans deux signaux modulés. C'est en contraste avec le principe des codes turbo où le bit d'information n'est utilisé qu'une seule fois. Pour éviter deux fois la transmission d'un groupe d'information k , un sélecteur est utilisé, il choisit alternativement un symbole du premier codeur puis du deuxième. Ce schéma permet ainsi d'atteindre une efficacité spectrale de k bits/s/Hz. Il fera l'objet d'une étude plus approfondie dans le chapitre suivant.

4.4 Schéma turbo pragmatique

Le schéma turbo pragmatique a été présenté par A. Glavieux, C. Berrou et S. Legoff [8] afin d'améliorer l'efficacité spectrale des codes turbo qu'ils avaient proposés. Le schéma général du turbo pragmatique est présenté à la figure 4.6.

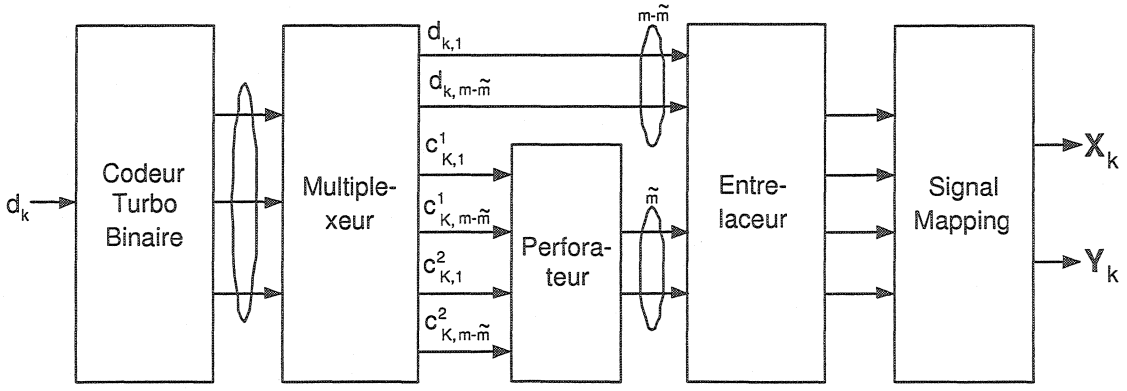

Figure 4.6: Codeur turbo pragmatique

Les sorties du codeur turbo binaire sont multiplexées et ensuite un module programmable perfore les symboles de parité pour obtenir $\tilde{m}$ symboles de parité et $m - \tilde{m}$ symboles systématiques. La perforation consiste à éliminer certains symboles. Le fait que la fonction de perforation soit programmable confère à ce schéma de modulation son caractère pragmatique. Il est ainsi aisé de modifier le taux de codage du codeur turbo et donc l'efficacité spectrale de la transmission, de plus en modifiant la constellation de la modulation on peut avoir une large gamme de schémas de modulations. Un entrelaceur est inséré avant le module d'assignation des bits à la constellation afin de décolérer les bits afin que le décodage turbo soit le plus efficace possible. Cependant, dans sa thèse, Legoff [32] montre que cette fonction n'est nécessaire que dans le cas d'un canal à evanouissement lent car elle n'apporte aucune amélioration de correction dans un canal gaussien.

Après la fonction d'entrelacement les bits sont ensuite assignés à des symboles multi-niveaux. La modulation utilisée peut être de type M-QAM ou M-PSK. L'assignation des bits à un signal de la constellation se fait selon un codage de Gray. Chaque signal de la constellation est représenté par un couple de valeurs réelles $(\mathbf{X}_k, \mathbf{Y}_k)$ codées par un ensemble de bit $d_{k,i}$, $i = 1...m$. Au niveau du codage de Gray, il apparaît que les bits permettant l'assignation d'un signal de la constellation ne sont pas tous protégés de la même manière. Cette assignation peut se faire de plusieurs manières selon qu'on protège d'avantage un bit de parité ou un bit systématique. Nous verrons par la suite que d'une assignation à une autre on peut avoir d'énormes variations de la qualité de transmission.

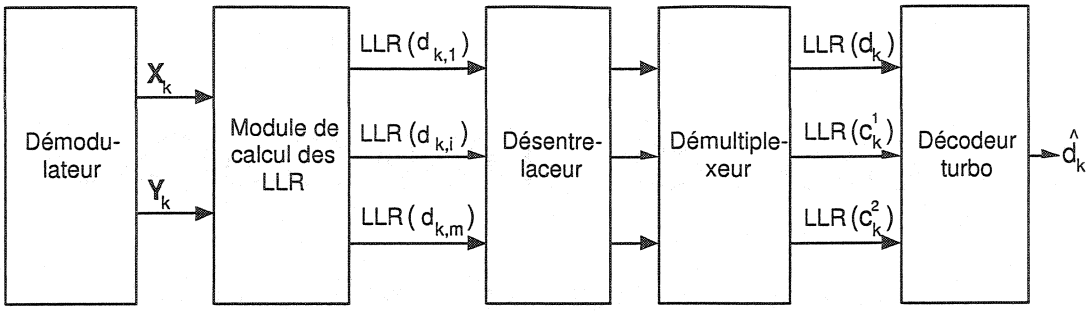

Figure 4.7: Décodeur turbo pragmatique

Le décodeur turbo pragmatique quant à lui se présente comme l'indique la figure 4.7 [8]. Le décodeur turbo utilisé est un décodeur binaire standard qui travaille non pas avec des symboles mais avec des bits. Il est donc nécessaire de retrouver les m bits qui ont servi à l'affectation d'un symbole de la constellation d'où la présence du module de calcul des LLR dont le rôle est l'estimation des bits qui ont constitué le symbole reçu. Etant obtenu à partir du même couple $(\mathbf{X}_k, \mathbf{Y}_k)$, ces m bits successifs sont fortement corréllés. Le désentrelaceur sert ainsi à les décoreller afin de garantir l'efficacité du décodage turbo. Les bits désentrelacés sont ensuite démultiplexés afin de retrouver les bits systématiques et les bits de parité. Le démultiplexeur introduit également des bits égales à zéro afin de remplacer ceux qui ont été supprimés lors de la perforation. Pour finir le décodeur turbo effectue son décodage turbo standard.

4.4.1 Etude du module de calcul des LLR

Il s'agit de déterminer à partir du signal reçu $\mathbf{R}_k$, une estimation des bits qui l'ont constitué. On effectue pour cela le calcul du LLR associé à chaque bit [8], [32].

$$LLR(d_{k,i}) = K_c \, Ln \frac{P(d_{k,i} = 1|\mathbf{R}_k)}{P(d_{k,i} = 0|\mathbf{R}_k)}, \quad i = 1 \ldots m. \tag{4.4}$$

où K_c est une constante. En appliquant la règle de Bayes et en supposant les bits consécutifs équiprobables, nous pouvons écrire:

$$LLR(d_{k,i}) = K_c Ln \frac{P((\mathbf{R}_k)|d_{k,i}=1)}{P((\mathbf{R}_k)|d_{k,i}=0)}, \quad i = 1 \ldots m. \tag{4.5}$$

A ce stade, il n'est plus possible de développer ces équations sans faire d'hypothèses sur le type de canal ainsi que sur la forme et le nombre de signaux de la constellation employée. Nous étudierons d'abord le cas d'un canal gaussien et ensuite celui d'un canal de Rayleigh.

4.4.1.1 Cas d'un canal gaussien

Dans cette sous section nous allons développer (4.5) en considérant plusieurs types de constellations. Le symbole p utilisé dans la suite est un entier.

*Cas d'une modulation (2^{2p})–QAM

Le signal reçu $\mathbf{R}_k$ se décompose en 2 signaux, le signal en phase $\mathbf{R}_{k,I}$ et le signal en quadrature $\mathbf{R}_{k,Q}$ correspondant au $k^{\text{ième}}$ symbole $(\mathbf{X}_k, \mathbf{Y}_k)$. L'expression du LLR devient :

$$LLR(d_{k,i}) = K_c Ln \frac{P((\mathbf{R}_{k,I}, \mathbf{R}_{k,Q})|d_{k,i}=1)}{P((\mathbf{R}_{k,I}, \mathbf{R}_{k,Q})|d_{k,i}=0)}, \quad i = 1 \ldots m. \tag{4.6}$$

Les signaux $\mathbf{R}_{k,I}$ et $\mathbf{R}_{k,Q}$ s'expriment en fonction du symbole transmis comme suit :

$$\begin{aligned} \mathbf{R}_{k,I} &= \mathbf{X}_k + n_{k,I} \\ \mathbf{R}_{k,Q} &= \mathbf{Y}_k + n_{k,Q} \end{aligned} \tag{4.7}$$

où $n_{k,I}$ et $n_{k,Q}$ sont 2 variables aléatoires gaussiennes non corrélées de moyennes nulles et de variance σ^2. Par conséquent $\mathbf{R}_{k,I}$ et $\mathbf{R}_{k,Q}$ sont également des variables aléatoires indépendantes gaussiennes, de moyennes respectives $\mathbf{X}_k$, $\mathbf{Y}_k$ et de variance σ^2. Le LLR peut se décomposer sous la forme:

$$\begin{aligned} LLR(d_{k,i}) &= K_c Ln \frac{P((\mathbf{R}_{k,I})|d_{k,i}=1)}{P((\mathbf{R}_{k,I})|d_{k,i}=0)}, \qquad i = 1 \ldots p. \\ &= K_c Ln \frac{P((\mathbf{R}_{k,Q})|d_{k,i}=1)}{P((\mathbf{R}_{k,Q})|d_{k,i}=0)}, \qquad i = (p+1) \ldots m. \end{aligned} \tag{4.8}$$

De manière plus explicite ces deux dernières expressions sont égales à:

$$LLR(d_{k,i}) = K_c * Ln \frac{\sum_{\mathbf{X}_k:d_{k,i}=1} exp(-\frac{(\mathbf{R}_{k,I}-\mathbf{X}_k)^2}{2\sigma^2})}{\sum_{\mathbf{X}_k:d_{k,i}=0} exp(-\frac{(\mathbf{R}_{k,I}-\mathbf{X}_k)^2}{2\sigma^2})}, \quad i = 1 \ldots p \tag{4.9}$$

et

$$LLR(d_{k,i}) = K_c * Ln \frac{\sum_{\mathbf{Y}_k:d_{k,i}=1} exp(-\frac{(\mathbf{R}_{k,Q}-\mathbf{Y}_k)^2}{2\sigma^2})}{\sum_{\mathbf{Y}_k:d_{k,i}=0} exp(-\frac{(\mathbf{R}_{k,Q}-\mathbf{Y}_k)^2}{2\sigma^2})}, \quad i = (p+1) \ldots m \tag{4.10}$$

Où $\mathbf{X}_k$ et $\mathbf{Y}_k$ sont les composantes en phase et en quadrature d'une modulation de signaux $d_{k,i} = j,\ j = 0, 1$. Ces équations présentent toutefois l'inconvénient d'être assez complexes à mettre en oeuvre parce qu'elles font intervenir beaucoup de calculs exponentiels. Pour $K_c = \sigma^2/2$, Legoff a mis en oeuvre une bonne approximation de ces équations [32] :

$$\begin{aligned} LLR(d_{k,1}) &= |\mathbf{R}_{k,I}| - 2^{p-1} \\ LLR(d_{k,i}) &= |LLR(d_{k,i-1})| - 2^{p-i},\ i = 2 \ldots (p-1) \\ LLR(d_{k,p}) &= \mathbf{R}_{k,I} \end{aligned} \tag{4.11}$$

et

$$\begin{aligned} LLR(d_{k,1+p}) &= |\mathbf{R}_{k,Q}| - 2^{p-1} \\ LLR(d_{k,i+p}) &= |LLR(d_{k,i-1+p})| - 2^{p-i},\ i = 2 \ldots (p-1) \\ LLR(d_{k,m}) &= \mathbf{R}_{k,Q} \end{aligned} \tag{4.12}$$

Malheureusement les LLR des bits $d_{k,1}$ à $d_{k,p}$ et ceux des bits $d_{k,p+1}$ à $d_{k,m}$, sont affectés par le bruit et ne sont pas indépendants. De plus les LLR($d_{k,i}$) ne sont pas gaussiens pour tous les bits, excepté pour $d_{k,p}$ et $d_{k,m}$. L'algorithme de décodage turbo ne peut donc pas s'appliquer puisqu'il nécessite que les données reçues soient des variables gaussiennes, voir chapitre 2. Néanmoins pour de larges rapports signal sur bruit les LLR se rapprochent d'une variable gaussienne de variance σ^2 [8].

*Cas d'une modulation 2^{2p+1}-QAM

Contrairement aux modulations QAM à 2^{2p} états, cette modulation ne peut pas se résumer à deux modulations d'amplitude indépendantes sur deux voies en quadrature. Par conséquent il est nécessaire de traiter conjointement les échantillons $\mathbf{R}_{k,I}$ et $\mathbf{R}_{k,Q}$, ce qui est défavorable vis à vis de la complexité de l'algorithme à implanter. Dans ce cas les $(2p+1)$ relations issues du calcul rigoureux de (4.5) sont les suivantes :

$$LLR(d_{k,i}) = K_c * Ln \frac{\sum_{\mathbf{X}_k:d_{k,i}=1} exp(-\frac{(\mathbf{R}_{k,I}-\mathbf{X}_k)^2}{2\sigma^2}) + \sum_{\mathbf{Y}_k:d_{k,i}=1} exp(-\frac{(\mathbf{R}_{k,Q}-\mathbf{Y}_k)^2}{2\sigma^2})}{\sum_{\mathbf{X}_k:d_{k,i}=0} exp(-\frac{(\mathbf{R}_{k,I}-\mathbf{X}_k)^2}{2\sigma^2}) + \sum_{\mathbf{Y}_k:d_{k,i}=0} exp(-\frac{(\mathbf{R}_{k,Q}-\mathbf{Y}_k)^2}{2\sigma^2})}$$

$$i = 1 \ldots 2p+1. \tag{4.13}$$

*Cas d'une modulation 2^m-PSK

Dans ce cas les signaux reçus sont:

$$\begin{aligned} \mathbf{R}_{k,a} &= cos\phi_k + n_{k,a} \\ \mathbf{R}_{k,b} &= sin\phi_k + n_{k,b} \end{aligned} \tag{4.14}$$

$cos\phi_k$ et $sin\phi_k$ sont les signaux transmis pour un symbole dont la phase est ϕ_k et $\mathbf{R}_{k,a}$, $\mathbf{R}_{k,b}$ les signaux respectifs reçus. $n_{k,a}$ et $n_{k,b}$ sont 2 variables aléatoires gaussiennes non corrélées de moyennes nulles et de variance σ^2. Par conséquent $\mathbf{R}_{k,a}$ et $\mathbf{R}_{k,b}$ sont également des variables aléatoires indépendantes gaussiennes, de moyennes respectives $cos\phi_k$ et $sin\phi_k$ et de variance σ^2. Le LLR s'exprime alors sous la forme :

$$LLR(d_{k,i}) = K_c * Ln\left[\frac{\sum_{cos\phi:d_{k,i}=1} exp(-\frac{(\mathbf{R}_{k,a}-cos\phi)^2}{2\sigma^2}) + \sum_{sin\phi:d_{k,i}=1} exp(-\frac{(\mathbf{R}_{k,b}-sin\phi)^2}{2\sigma^2})}{\sum_{cos\phi:d_{k,i}=0} exp(-\frac{(\mathbf{R}_{k,a}-cos\phi)^2}{2\sigma^2}) + \sum_{sin\phi:d_{k,i}=0} exp(-\frac{(\mathbf{R}_{k,b}-sin\phi)^2}{2\sigma^2})}\right]$$

$$i = 1 \ldots m. \tag{4.15}$$

4.4.1.2 Cas d'un canal de Rayleigh

Nous supposerons ici que le canal est sans mémoire et que l'atténuation α_k est connue du recepteur. Nous allons montrer dans ce cas qu'il n'est pas nécessaire de recommencer tout le raisonnement effectué pour un canal gaussien pour pouvoir trouver les estimations sur les bits qui ont constitués le symbole. Pour un canal de Rayleigh les signaux reçus $\mathbf{R}_{k,I}$ et $\mathbf{R}_{k,Q}$ s'expriment en fonction des signaux transmis $\mathbf{X}_k$ et $\mathbf{Y}_k$ comme suit :

$$\begin{aligned} \mathbf{R}_{k,I} &= \alpha_k \mathbf{X}_k + n_{k,I} \\ \mathbf{R}_{k,Q} &= \alpha_k \mathbf{Y}_k + n_{k,Q} \end{aligned} \tag{4.16}$$

La variable aléatoire α_k qui caractérise l'atténuation du signal émis suit une loi de probabilité de Rayleigh. Nous supposons connaître à priori α_k. Pour pourvoir

se ramener au cas gaussien nous allons considérer les deux variables $\mathbf{R}'_{k,I}$ et $\mathbf{R}'_{k,Q}$ définies comme suit :

$$\begin{aligned} \mathbf{R}'_{k,I} &= \frac{\mathbf{R}_{k,I}}{\alpha_k} = \mathbf{X}_k + n'_{k,I} \\ \mathbf{R}'_{k,Q} &= \frac{\mathbf{R}_{k,Q}}{\alpha_k} = \mathbf{Y}_k + n'_{k,Q} \end{aligned} \tag{4.17}$$

Avec :

$$\begin{aligned} n'_{k,I} &= \frac{n_{k,I}}{\alpha_k} \\ n'_{k,Q} &= \frac{n_{k,Q}}{\alpha_k} \end{aligned} \tag{4.18}$$

Les relations (4.17) sont similaires à celles obtenues pour un canal gaussien. $n'_{k,I}$ et $n'_{k,Q}$ sont deux bruits gaussiens de moyennes nulles et de variance $\overline{\sigma_k^2}$ égale à $\frac{\sigma^2}{\alpha_k^2}$. $\mathbf{R}'_{k,I}$ et $\mathbf{R}'_{k,Q}$ sont donc également deux variables aléatoires gaussiennes de variance σ_k^2 et de moyennes respectives $\mathbf{X}_k$ et $\mathbf{Y}_k$. Il est donc possible d'appliquer exactement le même algorithme que celui relatif au canal gaussien mais ici la variance ne sera plus σ^2 mais $\overline{\sigma_k^2}$. Au niveau du décodeur pragmatique il faudra introduire un module diviseur de signal par α_k juste avant le module de calcul des LLR.

4.4.2 Détermination de la variance du bruit gaussien

Les algorithmes de décodage turbo et ceux de l'estimation des bits utilisés pour la formation d'un symbole nécessitent la connaissance de la variance. La méthode que nous utiliserons pour déterminer la variance est celle utilisée par J. Parron [34]. On décide pour cela de transmettre des signaux d'énergie moyenne égale à 1. Soit R_g le taux de codage global du codeur, nous pouvons écrire :

$$mR_g * E_b = E_s = 1 \tag{4.19}$$

Nous avons d'autre part :

$$\begin{aligned} \sigma^2 &= \frac{N_0}{2} \\ &= \frac{1}{2}\frac{1}{E_s/N_0} \\ &= \frac{1}{2E_s/N_0} = \frac{1}{2mR_gE_b/N_0} \end{aligned} \tag{4.20}$$

où $m = log_2 M$. Nous utiliserons donc cette équation pour déterminer la variance à partir du rapport signal sur bruit E_b/N_0. Il faut remarquer que cette dernière formule s'applique à tout schéma de modulation multi-niveaux et donc aussi bien pour le schéma TTCM que pour le codeur multi-niveaux. Dans le cas d'une modulation BPSK, Reed et Asenstofer [38] ont quand à eux, proposé d'autres méthodes pour déterminer la variance. En fonction de la donnée systématique reçue, $x_{k,i}$ d'une séquence i de longueur N, on calcule cette variance à partir de la formule suivante.

$$\sigma_i^2 = \frac{\sum_{k=1}^{N} x_{k,i}^2}{N} - \left(\frac{\sum_{k=1}^{N} x_{k,i}}{N}\right)^2 \tag{4.21}$$

Ils ont également défini un autre critère qui utilise non seulement la séquence reçue mais aussi la séquence décodée, $\hat{d}_k$. La variance s'exprime comme suit :

$$\sigma_{i+1}^2 = \frac{\sum_{k=1}^{N}[x_{k,i} - (2\hat{d}_k - 1)]^2}{N} \tag{4.22}$$

4.4.3 Paramètres de simulations

Toutes les simulations qui effectuées dans ce chapitre seront faites pour un canal gaussien. Pour la fiabilité de toutes les simulations qui ont été faites dans ce mémoire, nous avons transmis un nombre de blocs N_{bl} qui nous permet d'avoir au moins 50 événements erreurs. D'une manière générale, on considère que 50 évènements erreurs constituent une bonne estimation de la probabilité d'erreur. La formule qui nous permet de calculer le nombre de blocs minimum est donnée par:

$$N_{bl} = \frac{50}{N.BER} \tag{4.23}$$

où N est la longueur de l'entrelaceur et BER la probabilité d'erreur estimée. Par exemple pour une probabilité d'erreur de 10^{-6}, et une longueur d'entrelaceur de $N = 5000$, nous devons transmettre au moins 10000 blocs de données.

Les simulations ont été faites pour des longueurs de contrainte K égales à 4 et 5 car elles nous permettent d'avoir le meilleur compromis entre la complexité et la performance d'erreur. Nous fournirons plus de détails dans la section 4.4.5.

Nous avons utilisé les meilleurs vecteurs générateurs, c'est à dire ceux qui maximisent la distance libre pour les CRS [55]. Ils sont présentés dans le tableau 4.1.

Selon l'étude effectuée par G. Royer [42], pour des longueurs d'entrelaceur N inférieures à 196, le meilleur compromis entre la complexité et la performance d'erreur serait l'entrelaceur uniforme et pour N supérieur à 196, ça correspond à l'entrelaceur aléatoire. Dans nos simulations nous utiliserons donc pour $N < 196$, l'entrelaceur uniforme et dans le cas contraire l'entrelaceur aléatoire.

Le taux de codage et l'ordre de la modulation sont toujours choisis de sorte à avoir une efficacité spectrale entière. Le tableau 4.2 résume toutes les configurations étudiées dans la suite du mémoire.

Tableau 4.1: Vecteurs générateurs des codes CRS constituant les codes turbo

Longueur de contrainte K	G_1	G_2
3	5	7
4	15	17
5	23	35
6	53	75
7	133	171
8	247	371
9	561	753

Tableau 4.2: Différentes configurations étudiées

Taux de codage	Constellation	Efficacité spectrale (bits/s/Hz)
1/2	16-QAM	2
2/3	8-PSK	2
3/4	16-QAM	3
4/5	32-QAM	4
2/3	64-QAM	4
5/6	64-QAM	5
3/4	256-QAM	6
7/8	256-QAM	7

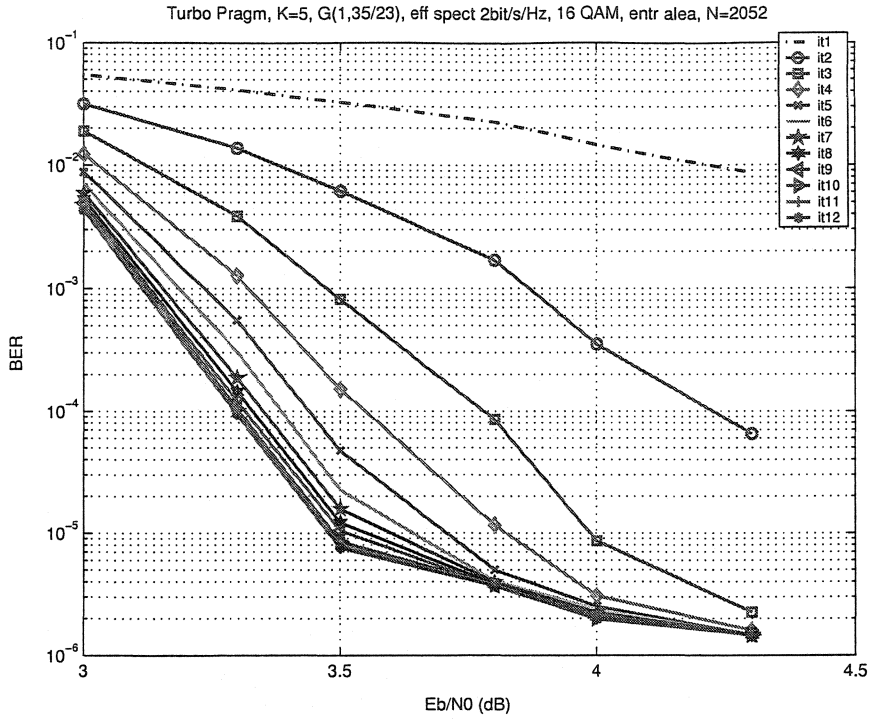

Figure 4.8: Influence du nombre d'itération sur les performances du décodeur turbo pragmatique sous un canal gaussien

Le décodage turbo permet d'obtenir des performances d'erreurs remarquables grâce à son décodage itératif [7]. Plus le nombre d'itérations augmente, meilleures sont ses performances. Malheureusement cette amélioration ne va pas sans une complexité supplémentaire. Il serait donc intéressant de voir quel est le nombre d'itérations qui nous permet d'avoir une bonne performance d'erreur sans toutefois nécessiter un énorme effort de calculs. Nous avons effectué des simulations en faisant varier le nombre d'itérations, tel qu'illustré à la figure 4.8. On constate qu'au delà de 6 itérations, on n'a pas un grand gain de codage, par exemple de l'itération 6 à l'itération 7 on a un gain d'à peine 0.03dB et de l'itération 6 à l'itération 12 on a un gain de 0.16 dB pour une probabilité d'erreur de 10^{-5}. Dans la suite du projet nous nous limiterons donc à 6 itérations pour effectuer notre décodage.

4.4.4 Etude de l'influence de l'assignation des bits aux symboles de la constellation [34]

Avec un codage de Gray, il apparaît que certains bits sont mieux protégés du bruit que d'autres. Nous verrons que d'une assignation à une autre on peut avoir une grande différence de gain de codage. Par exemple pour une probabilité d'erreur de 10^{-5} on peut avoir une variation allant jusqu'à 0.75 dB entre les assignations, voir la figure 4.12. Pour étudier l'influence de cette assignation nous allons considérer une modulation 16-QAM sous un canal gaussien. Un exemple de modulation 16-QAM est montré à la figure 4.9 où a est le facteur de normalisation qui nous permet de transmettre des signaux d'énergie égale à 1. Les bits $d_{k,1}$, $d_{k,2}$, $d_{k,3}$ et $d_{k,4}$ permettent d'assigner un symbole de la constellation

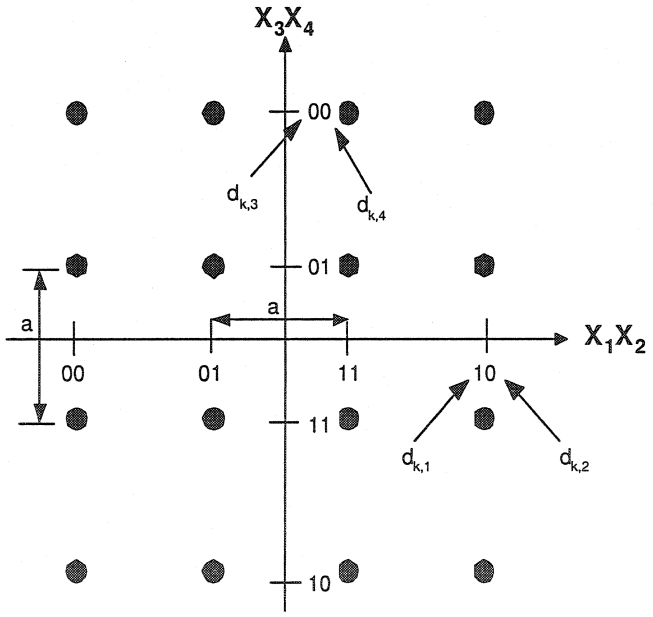

Figure 4.9: Constellation 16-QAM avec codage de Gray

Soit un taux de codage global 1/2. On dispose à la sortie du multiplexeur de 2 bits d'informations I1 et I2. A chacun de ces bits correspond les bits de parité du codeur convolutionnel 1 et 2. Nous disposons donc des bits de parité $P_{i,j}$: $P11$, $P12$, $P21$ et $P22$. i désigne l'indice du bit d'information et j celui du bit de parité.

Ces 4 bits de parité seront perforés pour n'en prendre que 2. Nous n'utiliserons que les bits $P11$ et $P22$ que nous noterons désormais $P1$ et $P2$. On aurait également pu utiliser $P12$ et $P21$ mais cela revient exactement au même. Les possibilités $(P11, P21)$ ou $(P12, P22)$ ne sont pas intéressantes parce qu'on utilisera seulement le bit de parité d'un seul CRS et on ne pourra pas effectuer le décodage turbo qui nécessite les bits de parité des deux CRS. Les bits d'informations $I1$, $I2$ et de partié $P1$, $P2$ seront utilisés pour former un symbole $S = (d_{k,1}, d_{k,2}, d_{k,3}, d_{k,4})$. Il existe plusieurs possibilités d'assignation à ces bits mais elles peuvent se résumer à six :

$$\begin{aligned} S_1 &= (I1P1\ I2P2) \\ S_2 &= (P1I1\ P2I2) \\ S_3 &= (P1I1\ I2P2) \\ S_4 &= (I1P2\ I2P1) \\ S_5 &= (P1I2\ P2I1) \\ S_6 &= (P1I2\ I1P2) \end{aligned} \tag{4.24}$$

On considère les cas où on privilégie les bits d'information sur les deux dimensions: les assignations S_1 et S_4, les bits de parité sur les deux dimensions : les assignations S_2 et S_5 et celui où on privilégie un bit de parité sur une dimension et un bit d'information sur l'autre : les assignations S_3 et S_6. Les trois premières assignations S_1, S_2 et S_3 correspondent aux cas où chaque bit d'information est associé avec son bit de parité sur une même dimension. Dans les trois dernières assignations S_4, S_5 et S_6, un bit d'information est associé avec le bit de parité de l'autre bit d'information.

Nous avions mentionné que les bits ne sont pas protégés de la même manière. Nous allons donc calculer les probabilités d'erreur sur une dimension de chacun des bits afin d'avoir une idée de la différence de protection obtenue et d'expliquer les résultats des simulations qui seront présentés dans les sections suivantes. Il n'est pas nécessaire de faire l'étude sur les deux dimensions d'autant plus qu'elles sont indépendantes. L'étude faite sur une dimension est également valable pour l'autre

dimension. Nous allons considérer les régions telles que décrites dans la figure 4.10. Sachant que l'on a émis le symbole $\mathbf{X}_I$ et reçu le symbole $\mathbf{R}_I$ nous avons :

$$\mathbf{R}_I = \mathbf{X}_I + n_I \tag{4.25}$$

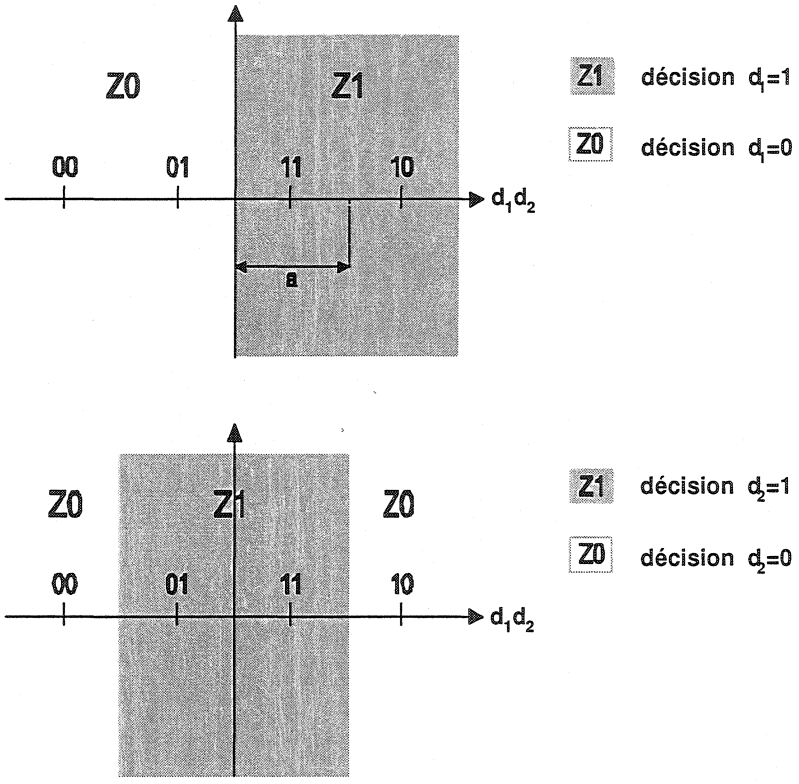

Figure 4.10: Régions de décision sur une dimension pour d_1 et d_2

Nous travaillons sur un canal gaussien, le bruit n_I est une variable aléatoire gaussienne de moyenne nulle et de variance σ^2. Calculons les probabilités d'obtenir les bits d_1 et d_2 en erreur.

$$P_\varepsilon(d_1) = P(\mathbf{R}_I \in Z1/d_1 = 0) * P(d_1 = 0) + P(\mathbf{R}_I \in Z0/d_1 = 1) * P(d_1 = 1) \tag{4.26}$$

Sachant que $P(d_1 = 0) = P(d_1 = 1) = \frac{1}{2}$ et étant donné la symétrie des régions de décision, nous pouvons écrire :

$$\begin{aligned} P_\varepsilon(d_1) &= P(\mathbf{R}_I \in Z1/d_1 = 0) \\ &= P(\mathbf{R}_I \in Z1/d_1 = 0, d_2 = 0) * P(d_2 = 0) \\ &\quad + P(\mathbf{R}_I \in Z1/d_1 = 0, d_2 = 1) * P(d_2 = 1) \\ &= \frac{1}{2} * \Big(P(n > \frac{3}{2}a) + P(n > \frac{1}{2}a) \Big) \\ &= \frac{1}{2} * \Big(\ Q(\frac{a}{2\sigma}) \ + \ Q(\frac{3a}{2\sigma}) \ \Big) \end{aligned} \tag{4.27}$$

Avec :

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty exp(-\frac{t^2}{2})dt \tag{4.28}$$

On procède de même pour d_2 :

$$\begin{aligned} P_\varepsilon(d_2) &= P(\mathbf{R}_I \in Z1/d_2 = 0) * P(d_2 = 0) \ + \ P(\mathbf{R}_I \in Z0/d_2 = 1) * P(d_2 = 1) \\ &= \frac{1}{2}\Big(P(\mathbf{R}_I \in Z1/d_2 = 0) + P(\mathbf{R}_I \in Z0/d_2 = 1) \Big) \end{aligned} \tag{4.29}$$

Or, du fait de la symétrie des régions de décision, nous avons :

$$\begin{aligned} P(\mathbf{R}_I \in Z1/d_2 = 0) &= P(\mathbf{R}_I \in Z1/x_2 = 0, x_1 = 0) * P(d_1 = 0) \\ &\quad + P(\mathbf{R}_I \in Z1/d_2 = 0, d_1 = 1) * P(d_1 = 1) \\ &= P(\mathbf{R}_I \in Z1/d_2 = 0, d_1 = 0) \end{aligned} \tag{4.30}$$

De la même façon, nous pouvons écrire :

$$P(\mathbf{R}_I \in Z0/d_2 = 1) = P(\mathbf{R}_I \in Z0/d_2 = 1, d_1 = 0) \tag{4.31}$$

D'où :

$$
\begin{aligned}
P_e(d_2) &= \frac{1}{2} * \Big(P(\mathbf{R}_I \in Z1/d_2 = 0, d_1 = 0) + P(\mathbf{R}_I \in Z0/d_2 = 1, d_1 = 0) \Big) \\
&= \frac{1}{2} * \Big(P(n > \frac{a}{2\sigma}) + \big[P(n < -\frac{a}{2\sigma}) + P(n > \frac{3a}{2\sigma}) \big] \Big) \\
&= \frac{1}{2} \Big(\ 2 * Q(\frac{a}{2\sigma}) \ + \ Q(\frac{3a}{2\sigma}) \ \Big) \qquad (4.32)
\end{aligned}
$$

(4.27) et (4.32) montrent que la probabilité d'avoir les bits d_1 et d_3 en erreur est deux fois plus faible à celle d'avoir d_2 et d_4 en erreur. Les bits d_1 et d_3 sont donc deux fois mieux protégés que les bits d_2 et d_4. C'est donc le premier bit de la dimension qui est le plus protégé. Par conséquent utiliser l'assignation ($I1P1I2P2$) revient à privilégier les bits d'informations par rapport aux bits de parité. La valeur de σ est calculée à partir de (4.20). Nous avons vu que pour que (4.20) soit valide il faudrait que l'énergie moyenne de la constellation soit égale à 1. Il faudra donc normaliser cette énergie avant l'émission d'un symbole. Lorsque la modulation utilisée est de la forme M-QAM les signaux en phase et en quadrature sont multipliés par un coefficient a qui permet de normaliser la constellation. Le tableau 4.3 récapitule les différentes valeurs du coefficient a pour différents ordres de modulations. Pour une modulation 16-QAM, par exemple l'énergie totale de la constellation est égale à : $40a^2$, la valeur de a sera donc de $\sqrt{2/5}$ pour avoir une énergie moyenne de la constellation égale à 1. Pour une modulation M-PSK les signaux sont déjà normalisés car les signaux transmis sont de la forme $cos\phi$ et $sin\phi$.

Tableau 4.3: Coéfficient de normalisation

Modulation M-QAM	a^2
16 QAM	2/5
32 QAM	2/13
64 QAM	2/21
128 QAM	2/53
256 QAM	2/85

4.4.4.1 Résultats de simulation

Les résultats des simulations pour les six assignations sont présentés aux figures 4.11 à 4.16. Ces courbes ont été effectuées pour une modulation 16-QAM, un taux de codage $R = 1/2$ et donc pour une efficacité spectrale de 2 bits/s/Hz. Il s'agit de voir lesquelles des six assignations sont les meilleures et ce pour des longueurs de contrainte et d'entrelaceur différentes.

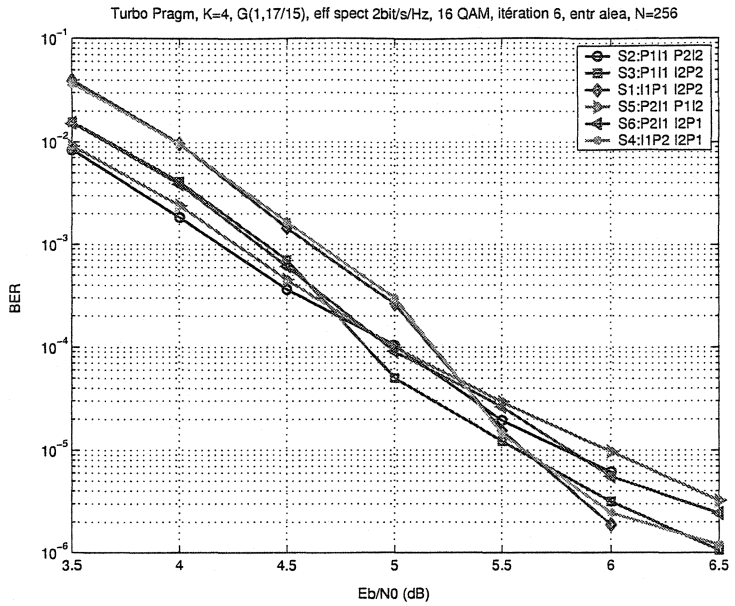

Figure 4.11: Influence de l'assignation sur les performances du codeur turbo pragmatique, $K = 4$, $N = 256$, efficacité spectrale 2 bits/s/Hz, 16-QAM

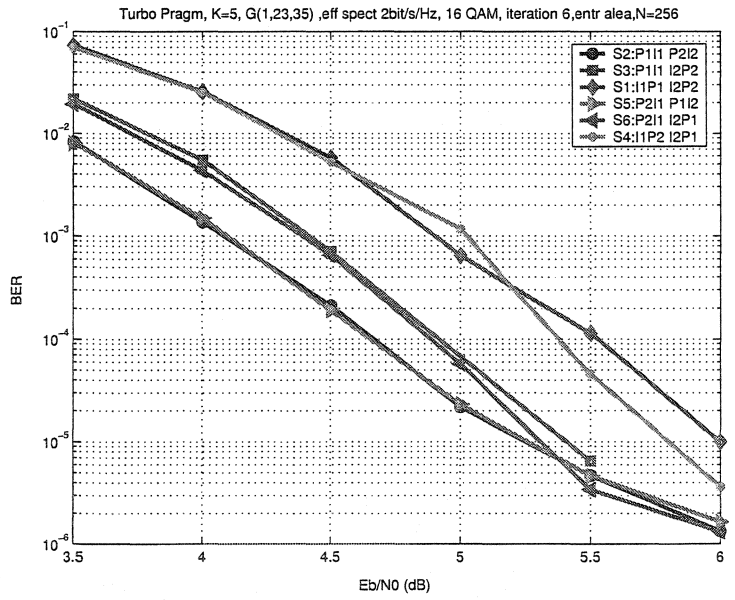

Figure 4.12: Influence de l'assignation sur les performances du codeur turbo pragmatique, $K = 5$, $N = 256$, efficacité spectrale 2 bits/s/Hz, 16-QAM

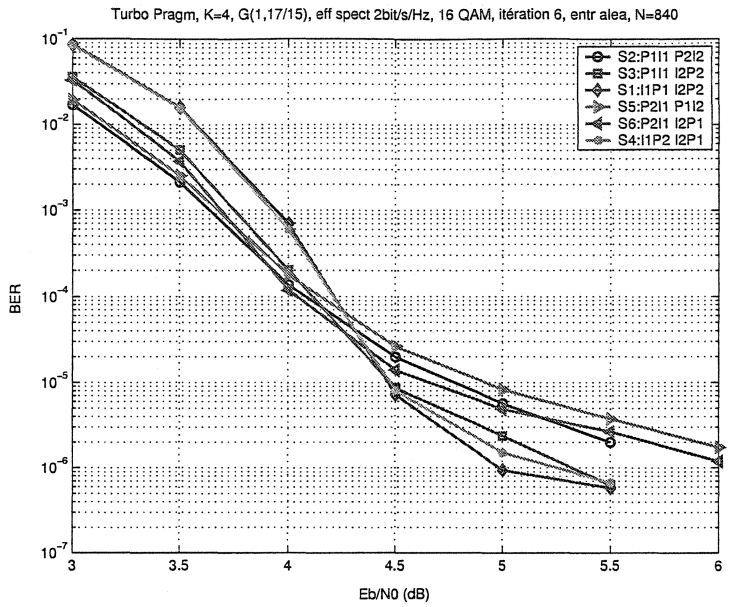

Figure 4.13: Influence de l'assignation sur les performances du codeur turbo pragmatique, $K = 4$, $N = 840$, efficacité spectrale 2 bits/s/Hz, 16-QAM

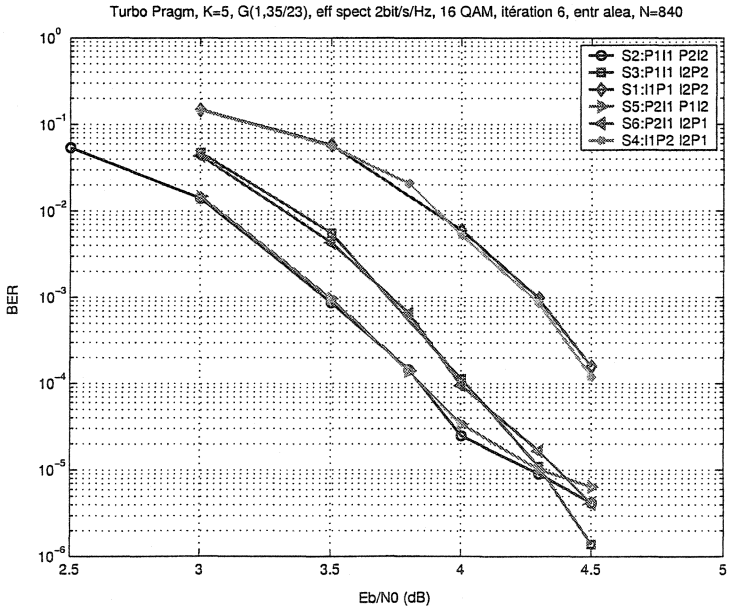

Figure 4.14: Influence de l'assignation sur les performances du codeur turbo pragmatique, $K = 5$, $N = 840$, efficacité spectrale 2 bits/s/Hz, 16-QAM

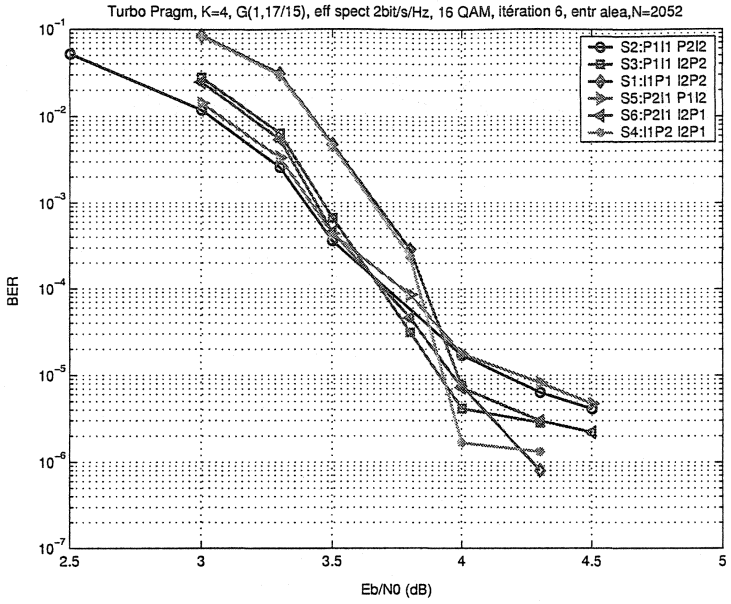

Figure 4.15: Influence de l'assignation sur les performances du codeur turbo pragmatique, $K = 4$, $N = 2052$, efficacité spectrale 2 bits/s/Hz, 16-QAM

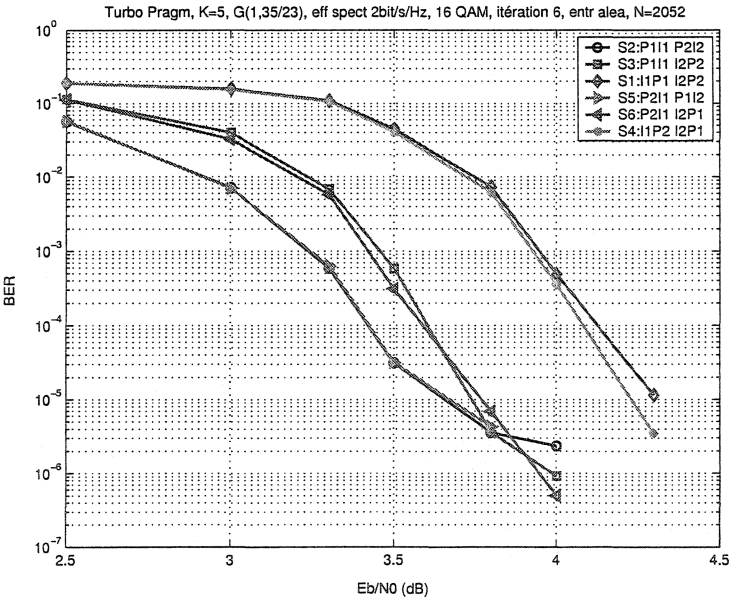

Figure 4.16: Influence de l'assignation sur les performances du codeur turbo pragmatique, $K = 5$, $N = 2052$, efficacité spectrale 2 bits/s/Hz, 16-QAM

On constate que quelque soit la longueur de contrainte K les courbes obtenues tendent à se démarquer deux à deux. Le fait d'associer sur une dimension une information appartenant à deux codeurs CRS différents ou bien au même codeur CRS n'apporte pratiquement aucune différence. C'est tout à fait normal, parce qu'en fait ce qui importe c'est de privilégier un bit de parité ou d'information. Les deux dimensions sont indépendantes et se comporte pareillement, qu'une information d'un codeur CRS se retrouve avec l'information de l'autre codeur CRS ou de lui même n'a aucune importance. Dans la suite du chapitre nous nous limiterons donc désormais à l'étude de trois assignations d'autant plus qu'en fait étudier les six assignations revient en fait à en n'étudier que trois: $S_1 = (I1P1\ I2P2)$, $S_2 = (P1I1\ P2I2)$ et $S_3 = (P1I1\ I2P2)$.

Pour $K = 4$, on remarque que l'assignation S_1 tend à saturer beaucoup plus vite et sa pente est moins abrute que celle de l'assignation S_2. L'assignation S_2 présente de meilleures performances que les assignations S_1 et S_3 pour de faibles rapport signal sur bruit et pour une probabilité d'erreur supérieure à $5 * 10^{-5}$. L'inverse se produit pour une probabilité d'erreur inférieure, l'assignation S_1 présente de meilleures performances d'erreur. Pour une probabilité d'erreur recherchée de 10^{-4} par exemple on choisira l'assignation S_2 et pour un BER $< 10^{-5}$, on choisira l'assignation S_1.

Pour $K = 5$ par contre, ce phénomène se produit à partir d'une probabilité d'erreur beaucoup plus faible de l'ordre de 10^{-5}. L'assignation S_2 sera donc toujours la meilleure étant donné qu'on ne recherche pratiquement jamais à atteindre une telle probabilité d'erreur. Des résultats intermédiaires sont obtenus pour S_3, où on privilégie les bits de parité sur une dimension et un bit d'information sur l'autre et ce pour les deux longueurs de contrainte.

On remarque d'autre part que l'écart entre les assignations est beaucoup plus accentué pour une longueur de contrainte $K = 5$ par rapport à $K = 4$. C'est tout à fait normal parce qu'étant donné que les bits de parité traduisent la puissance du codeur, leur influence se fera moins sentir lorsque K augmente. On se retrouve

avec des gains de codage de 0.75 et de 0.35dB de l'assignation S_1 à S_2 pour des valeurs de $K = 5$ et de $K = 4$, respectivement.

Plusieurs autres facteurs peuvent influencer les performances du turbo pragmatique notamment la longueur de contrainte des CRS, la longueur de l'entrelaceur, l'efficacité spectrale. Dans les sections qui suivent, nous étudierons l'influence de ces facteurs sur l'influence de l'assignation des bits à la constellation et sur la performance d'erreur.

4.4.5 Influence de la longueur de contrainte K du code

Il est évident que la complexité du décodage augmente avec la longueur de contrainte. Il s'agit de déterminer la longueur de contrainte qui offre un bon compromis entre la performance d'erreur et la complexité de décodage. Afin d'examiner l'influence de la longueur de contrainte CRS sur le pouvoir de correction du codeur turbo pragmatique, nous avons effectué des simulations pour des longueurs de contrainte K allant de 3 à 7. Les simulations ont été effectuées pour 4 longueurs d'entrelaceur différentes : $N = 256$, 840, 2052 et 4096. Les résultats obtenus sont présentés sur les figures 4.17 à 4.20.

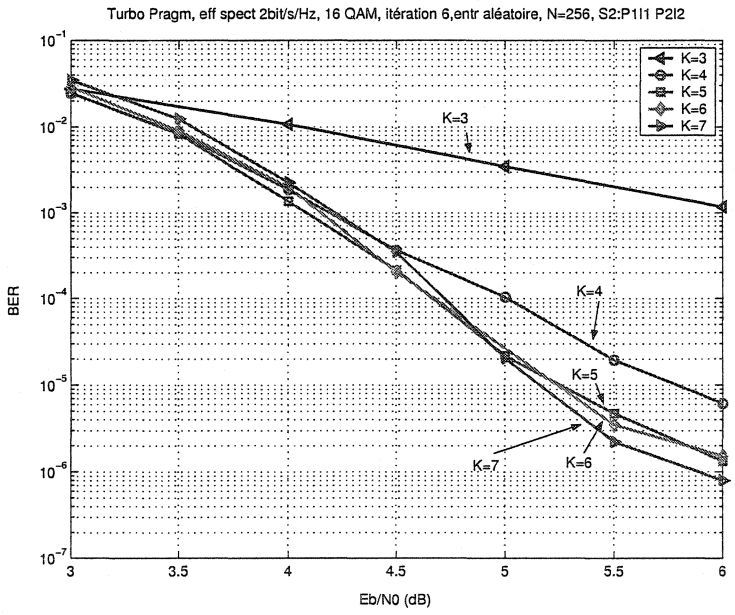

Figure 4.17: Influence de la longueur de contrainte, codeur turbo pragmatique, $N = 256$, itération 6, 16-QAM

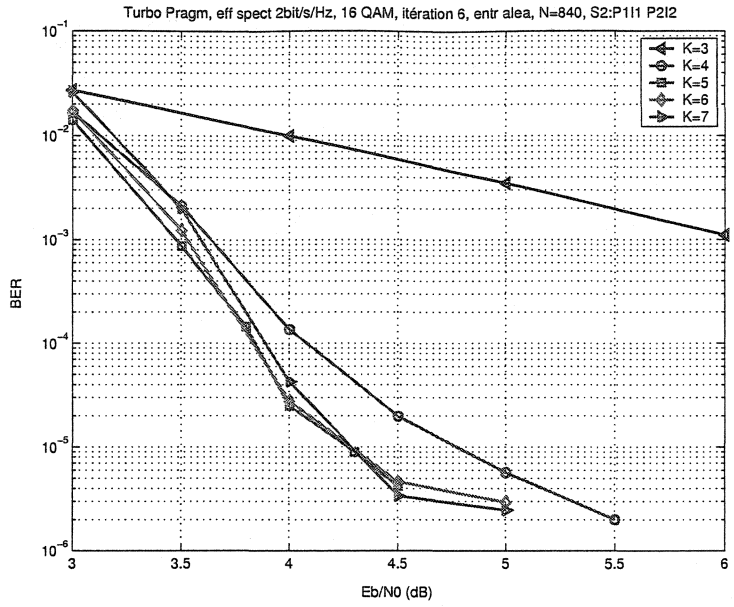

Figure 4.18: Influence de la longueur de contrainte, codeur turbo pragmatique, $N = 840$, itération 6, 16-QAM

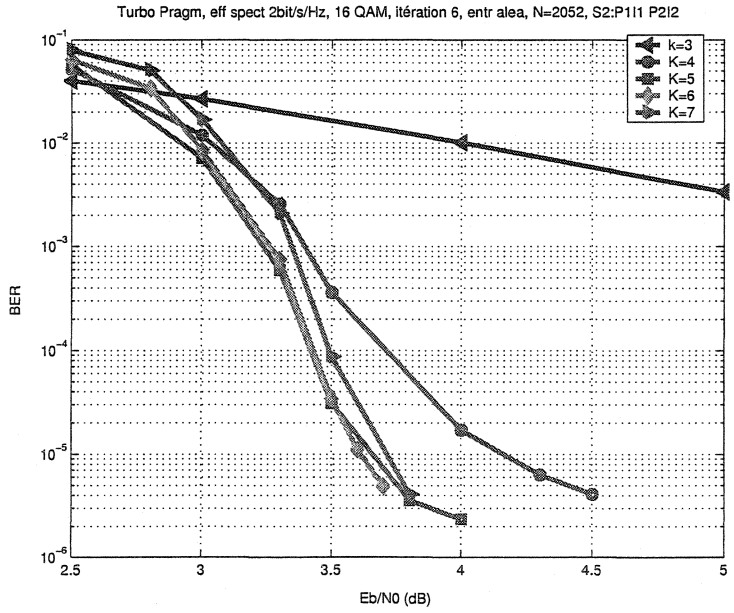

Figure 4.19: Influence de la longueur de contrainte, codeur turbo pragmatique, $N = 2052$, itération 6, 16-QAM

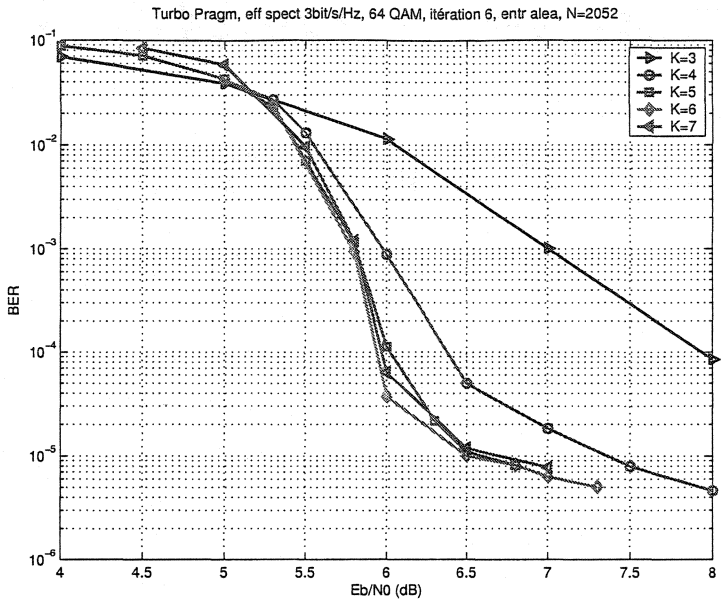

Figure 4.20: Influence de la longueur de contrainte, codeur turbo pragmatique, $N = 2052$, itération 6, 64-QAM

On constate que quelque soit l'efficacité spectrale, le nombre d'itération et la longueur d'entrelaceur utilisés (N=256, 840, 2052 ou 4096), on retrouve les mêmes caractéristiques. Le code de longueur de contrainte $K = 3$ ne dispose pas d'une capacité de correction suffisante pour présenter un intérêt. Seules les longueurs de contrainte 4 et 5 présentent un excellent compromis entre la performance d'erreur et la complexité. Avec $K = 5$ nous obtenons un léger gain, de 0.6 à 0.8 dB par rapport à $K = 4$ pour une probabilité d'erreur 10^{-5}. Au delà de 5, pour $K = 6$ et $K = 7$, on n'a aucune amélioration de la performance d'erreur, ces deux longueurs de contrainte ne présentent donc aucun interêt. Pour essayer de comprendre ces résultats nous examinons les travaux qu'ont effectué Benedetto [6], Divsalar [15] et Perez [35] sur le spectre de distance des codes turbo. La borne union de la probabilité d'erreur pour les codes turbo est donnée par [6] :

$$P_b \leq \sum_{d=d_{free}}^{\infty} \sum_{w=1}^{N} (n_w.w) / \binom{N}{w} . P_d \tag{4.33}$$

où n_w est le nombre de mots de code de poids w, N la longueur de la séquence d'information et P_d, la probabilité d'erreur d'une paire de mots de code de distance égale à d, P_d est donnée par l'expression:

$$P_d = Q\left(\sqrt{2.d.R.\frac{E_b}{N_0}}\right) \tag{4.34}$$

On remarque que la distance libre intervient dans la détermination de la borne union sur la probabilité d'erreur. La détermination de cette distance libre semble toutefois assez complexe à déterminer. Divsalar [15] a montré que pour un canal gaussien les performances d'erreur des codes turbo sont largement déterminées par un poids d'entrée $w = 2$. Soit $z_{\min}$ le poids de Hamming d'un mot de code pour une séquence d'entrée à coder de poids w=2. Il a défini la distance effective libre $d_{\text{free,eff}}$ d'un code turbo comme :

$$d_{\text{free,eff}} = 2 + 2z_{\min} \tag{4.35}$$

Les performances d'erreurs seront plutôt essentiellement déterminées par la distance effective libre que par la distance libre pour les codes convolutionnels, surtout si on utilise un entrelaceur aléatoire. Divsalar a déterminé la distance effective libre pour différentes longueurs de contrainte [15]. Pour $K = 5$, il a trouvé une distance effective libre de 10 et de 11 pour $K = 6$ et $K = 7$. Le fait qu'il n'y ait pas une grande différence de la distance effective libre entre ces longueurs de contrainte expliquerait les résultats des simulations obtenus pour ces 3 longueurs de contrainte.

4.4.6 Influence de l'efficacité spectrale

Les schémas de modulations multi-niveaux tiennent leur importance du fait qu'ils permettent d'avoir des efficacités spectrales élévées nécessaires pour les applications à hauts débits dans des largeurs de bande réduite. On ne saurait donc parler de turbo pragmatique sans étudier l'influence de l'efficacité spectrale sur les performances d'erreur et sur les assignations que nous avons retenues. Nous avons effectué plusieurs simulations en faisant varier l'efficacité spectrale de 2 bits/s/Hz à 7 bits/s/Hz. Quelques résultats sont présentés dans les figures 4.21 à 4.26. Ces résultats ont été obtenus pour une longueur d'entrelaceur N égale à 2052, pour l'assignation S_1 et pour un code de longueur de contrainte $K = 5$.

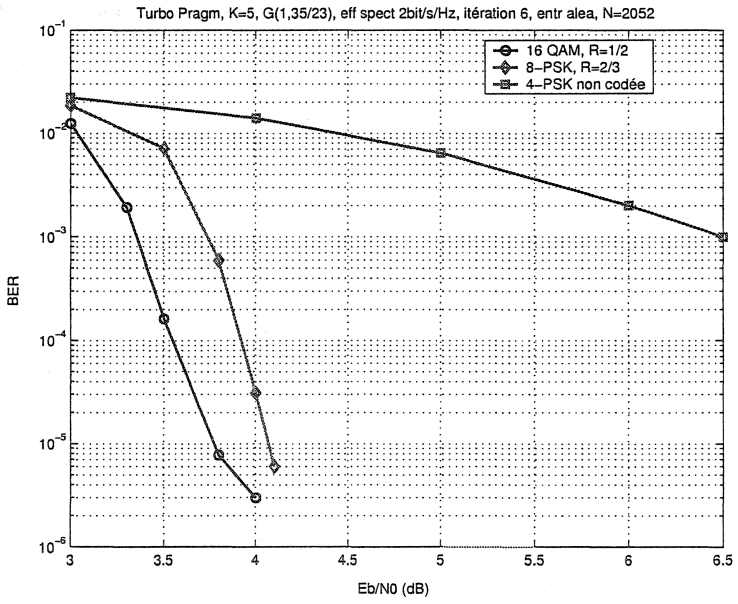

Figure 4.21: Codeur turbo pragmatique, $K = 5$, $G = (1, 35/23)$, efficacité spectrale 2 bits/s/Hz

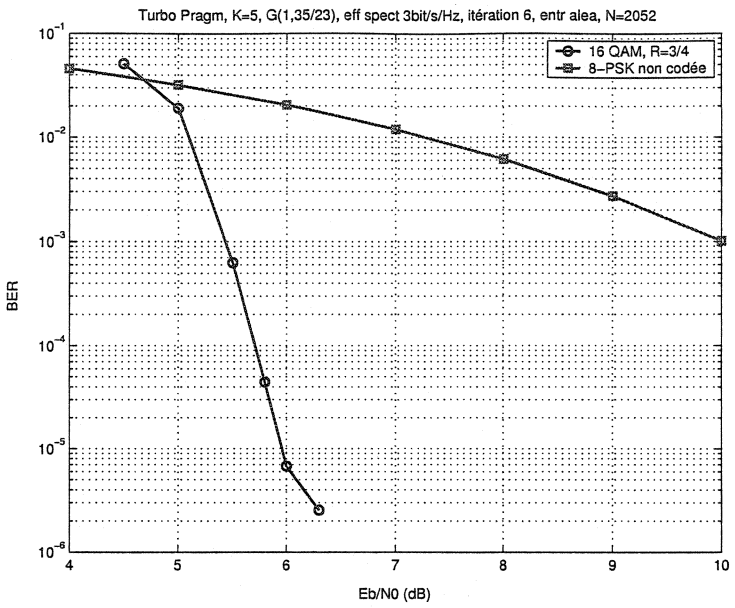

Figure 4.22: Codeur turbo pragmatique, $K = 5$, $G = (1, 35/23)$, efficacité spectrale 3 bits/s/Hz

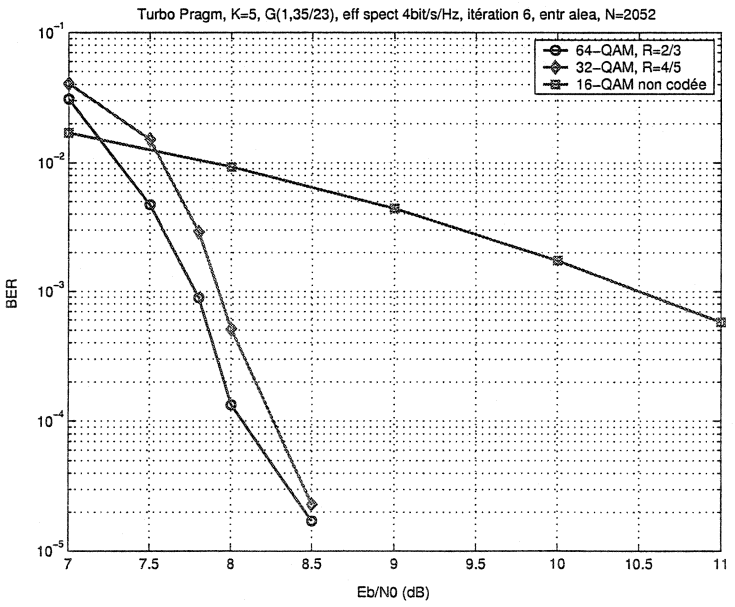

Figure 4.23: Codeur turbo pragmatique, $K = 5$, $G = (1, 35/23)$, efficacité spectrale 4 bits/s/Hz

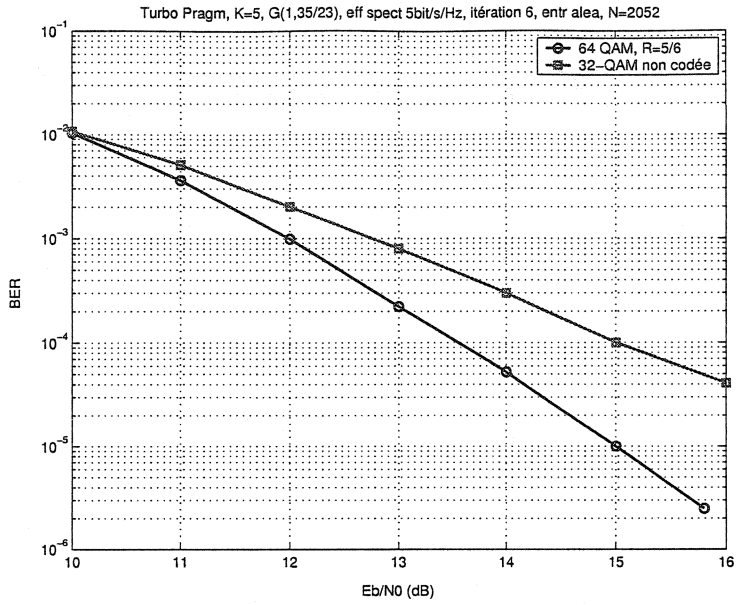

Figure 4.24: Codeur turbo pragmatique, $K = 5$, $G = (1, 35/23)$, efficacité spectrale 5 bits/s/Hz

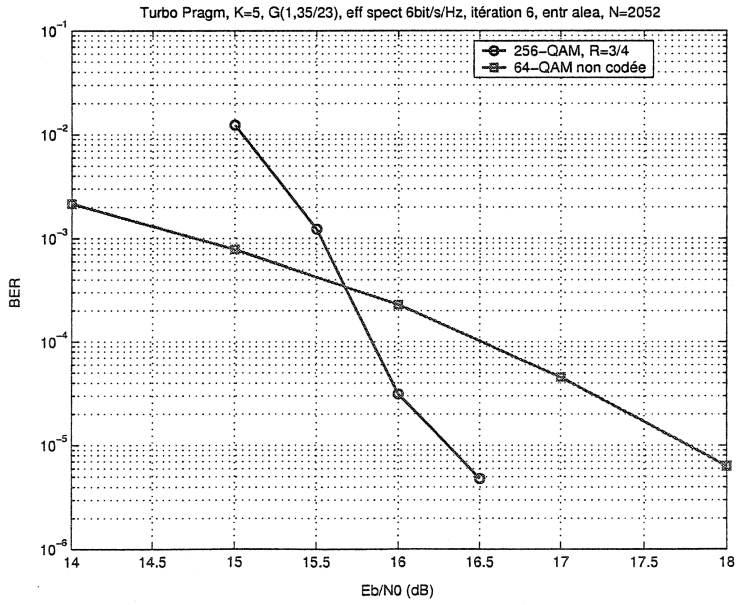

Figure 4.25: Codeur turbo pragmatique, $K = 5$, $G = (1, 35/23)$, efficacité spectrale 6 bits/s/Hz

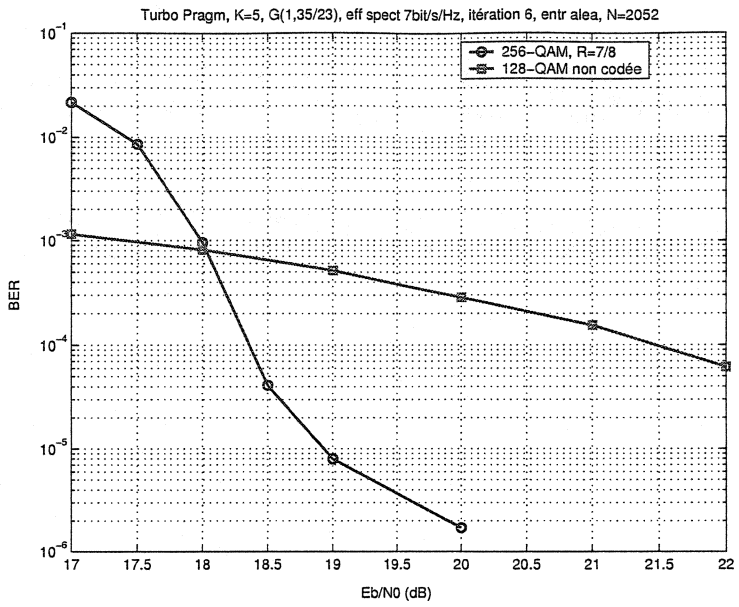

Figure 4.26: Codeur turbo pragmatique, $K = 5$, $G = (1, 35/23)$, efficacité spectrale 7 bits/s/Hz

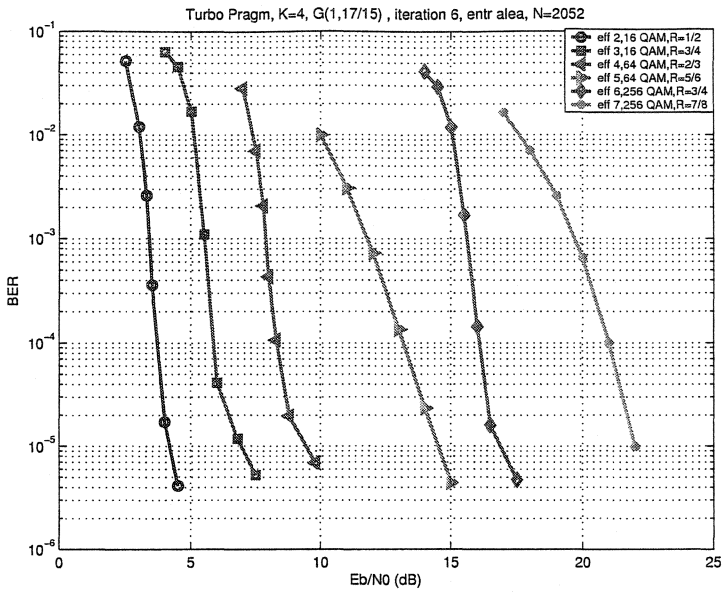

Figure 4.27: Performances du codeur turbo pragmatique pour différentes efficacités spectrales, $K = 4$, $G = (1, 17/15)$, $N = 2052$

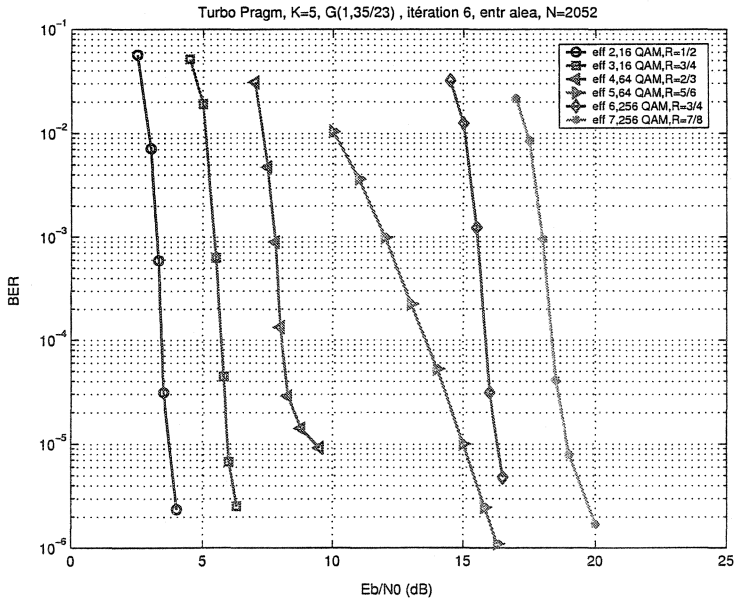

Figure 4.28: Performances du codeur turbo pragmatique pour différentes efficacités spectrales, $K = 5$, $G = (1, 35/23)$, $N = 2052$

On remarque que pour des constellations qui donnent la même efficacité spectrale on a à peu près les mêmes performances d'erreurs à 0.2dB de différence à 10^{-5}: voir les figures 4.21 et 4.23, que ce soit pour une longueur de contrainte $K = 4$ ou $K = 5$. les performances d'erreur s'empirent lorsque l'efficacité spectrale augmente. Le passage d'une efficacité spectrale à sa valeur supérieure demande un rapport signal sur bruit $Eb/N0$ en moyenne de 2.7dB pour une probabilité d'erreur de 10^{-5}: voir les figures 4.27 et 4.28. Ces deux dernières courbes recensent les simulations obtenues pour une efficacité spectrale allant de 2 à 7. On obtient des gains de codage d'environ 5 dB pour une probabilité d'erreur de 10^{-5} par rapport à la modulation non codée qui nous aurait permis d'avoir la même efficacité spectrale excepté pour une efficacité spectrale de 3 et 6 bits/s/Hz, voir les figures 4.22 et 4.25.

Dans l'introduction de ce mémoire, il a été mentionné que selon les travaux de Shannon on pouvait transmettre de l'information avec une probabilité d'erreur nulle à condition que le taux de transmission R soit inférieur à la capacité du canal C donnée par (4.36) pour un canal gaussien:

$$C = Wlog_2\Big(1 + \frac{P}{NoW}\Big) \;\; bits/s \tag{4.36}$$

(4.36) peut se réécrire sous les formes suivantes :

$$\begin{aligned} C &= Wlog_2\Big(1 + \frac{C.E_b}{NoW}\Big) \\ \frac{C}{W} &= log_2\Big(1 + \frac{C}{W}\frac{E_b}{No}\Big) \end{aligned} \tag{4.37}$$

L'efficacité spectrale η étant définie comme le rapport entre la capacité et la largeur de bande, (4.37) peut se réécrire comme suit :

$$\eta = log_2\Big(1 + \eta\frac{E_b}{No}\Big) \tag{4.38}$$

Le rapport signal sur bruit est donc égale à:

$$\frac{E_b}{No} = \frac{2^\eta - 1}{\eta} \tag{4.39}$$

$$\left(\frac{E_b}{No}\right)_{dB} = 10log_{10}\left(\frac{2^\eta - 1}{\eta}\right) \tag{4.40}$$

Soit M l'ordre de la modulation utilisée et R, le taux de codage du codeur global. L'efficacité spectrale η est donnée par : $\eta = Rlog_2M$. En introduisant cette formule dans (4.39), nous obtenons :

$$\left(\frac{E_b}{No}\right)_{dB} = 10log_{10}\left(\frac{2^{mR} - 1}{mR}\right) \tag{4.41}$$

En se servant de cette dernière équation, nous pouvons calculer la limite de Shannon pour un taux de codage et un ordre de modulation donnés. Si l'on choisit comme la plupart des auteurs une valeur de référence de 10^{-5} pour la probabilité d'erreur par bit (BER), on peut alors calculer l'écart entre la limite de Shannon et les différents BER obtenus, (voir tableau 4.4).

Tableau 4.4: Ecart entre la limite théorique de Shannon et le turbo pragmatique pour $N = 2052$ et $K = 5$

Taux de codage R	Ordre de la modulation	Limite de Shannon (dB)	E_b/No pour BER $= 10^{-5}$ (dB)	Ecart par rapport à la limite de Shannon (dB)
1/2	16-QAM	1.76	3.63	1.87
2/3	8-PSK	1.76	4.1	2.34
3/4	16-QAM	3.67	6.0	2.33
4/5	32-QAM	5.74	8.6	2.86
2/3	64-QAM	5.74	8.6	2.86
5/6	64-QAM	7.92	15.0	7.08
3/4	256-QAM	10.21	16.3	6.09
7/8	256-QAM	12.58	18.7	6.12

Nous remarquons que plus l'efficacité spectrale augmente et plus on s'éloigne de la limite théorique de Shannon. En effet l'écart entre la limite théorique de Shannon et la limite pratique est d'environ 2dB pour une efficacité spectrale de 2 bits/s/Hz, de 2.33dB pour 3 bits/s/Hz, 2.86 pour 4 bits/s/Hz et de près de 6dB pour une efficacité spectrale supérieure à 4.

Nous avons également recensé dans le tableau 4.5 les gains de codage obtenus pour plusieurs efficacités spectrales par rapport au système non codé qui nous aurait permis d'avoir la même efficacité spectrale.

Tableau 4.5: Gain de codage par rapport à la modulation non codée à 10^{-5} pour $K = 5$ et $N = 2052$

Taux de codage R	Ordre de la modulation	Gain à 10^{-5} par rapport à la modulation non codée (dB)
1/2	16-QAM	5.5
2/3	8-PSK	5.03
3/4	16-QAM	7.5
4/5	32-QAM	5.0
2/3	64-QAM	5.0
5/6	64-QAM	2.6
3/4	256-QAM	1.6
7/8	256-QAM	4.9

4.4.7 Influence de l'efficacité spectrale sur l'assignation des bits à la constellation

Il serait intéressant de voir l'influence que peut avoir l'efficacité spectrale sur l'assignation des bits à la constellation. Nous avons effectué pour cela plusieurs simulations qui sont présentés dans les figures 4.29 à 4.31.

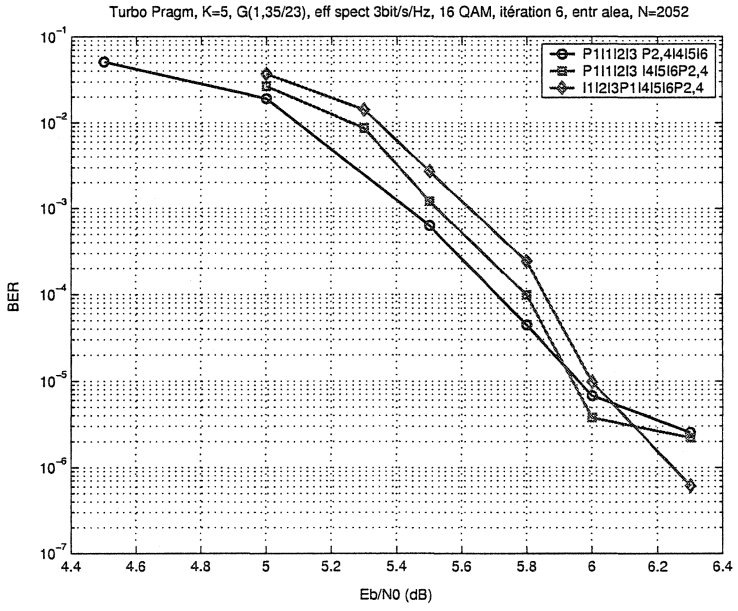

Figure 4.29: Influence de l'efficacité sur l'assignation des bits, $K = 5$, $G = (1, 35/23)$, efficacité spectrale 3 bits/s/Hz

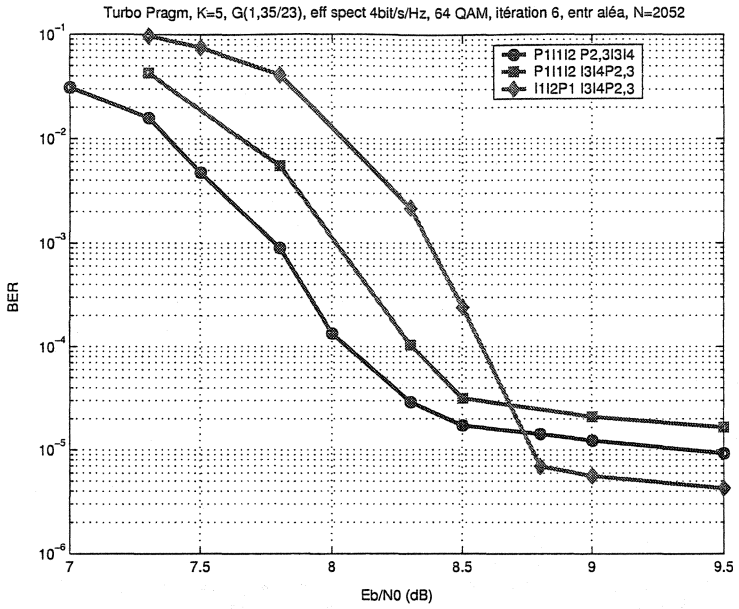

Figure 4.30: Influence de l'efficacité sur l'assignation des bits, $K = 5$, $G = (1, 35/23)$, efficacité spectrale 4 bits/s/Hz

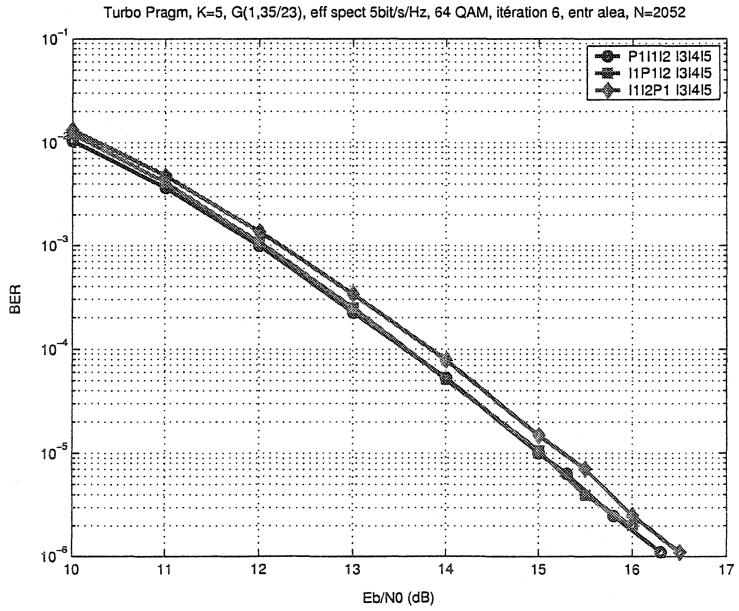

Figure 4.31: Influence de l'efficacité sur l'assignation des bits, $K = 5$, $G = (1, 35/23)$, efficacité spectrale 5 bits/s/Hz

Il apparaît clairement que l'influence de l'assignation diminue nettement avec l'augmentation de l'efficacité spectrale pour un même type de constellation. Voir les courbes 4.16 à 4.29, 4.30 à 4.31 et II.8 à II.9. Pour une efficacité spectrale égale à 5 bits/s/Hz, il n'y a plus aucune différence. Lorsque l'efficacité spectrale augmente, le nombre de bits de parités envoyé diminue par rapport au nombre de bit d'information et son influence se fait moins sentir. Cette parité n'a plus un grand effet sur la performance d'autant plus que l'effet qu'elle produit est noyé devant celui des bits d'information. Pour un rapport k, (nombre de bit de parité sur nombre de bit d'information), égal à $1/5$ il n'y plus aucune différence : voir la figure 4.31. On constate également que l'écart est accentué pour $K = 5$ par rapport à $K = 4$, voir les figures II.7 à II.9 et 4.29 à 4.31. Il est tout à fait normal que, lorsque K augmente, son influence se fasse plus sentir dans l'influence de l'assignation en fonction de l'efficacité spectrale étant donné que les bits de parité traduisent la puissance du codeur.

4.4.8 Influence de la longueur N de l'entrelaceur

La taille de l'entrelaceur a sans aucun doute une influence sur les performances des codes turbo. Dans cette section nous nous sommes intéressés à l'influence que pouvait avoir cette longueur sur les performances du schéma turbo pragmatique. Nous avons effectué des simulations dont les résultats sont présentés dans les figures 4.32 à 4.33.

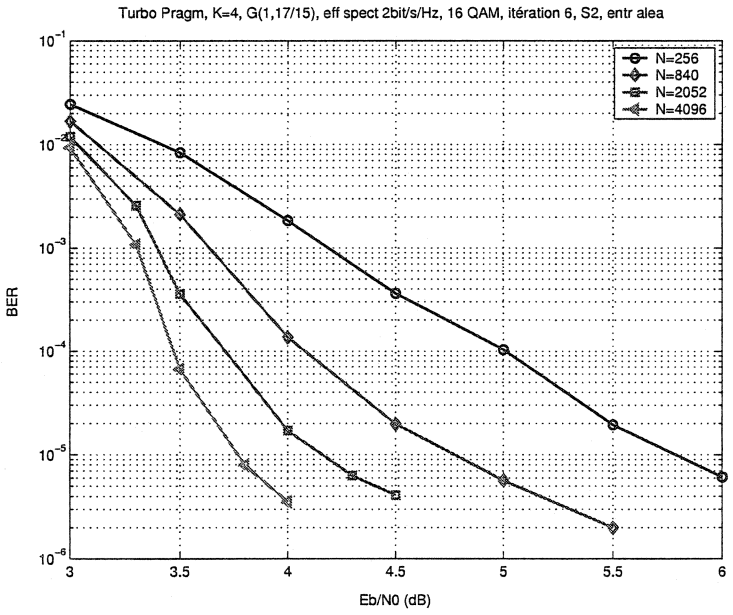

Figure 4.32: Influence de la longueur de l'entrelaceur sur les performances du codeur turbo pragmatique, $K = 4$, $G = (1, 17/15)$, S_2

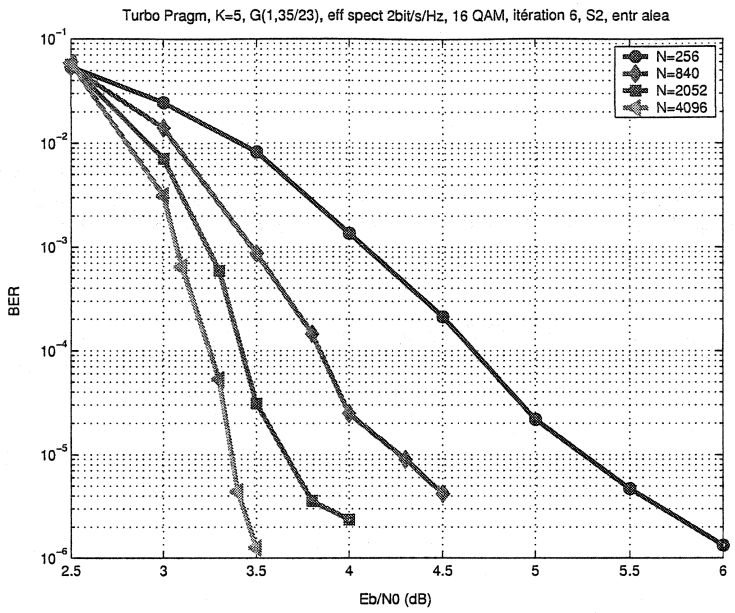

Figure 4.33: Influence de la longueur de l'entrelaceur sur les performances du codeur turbo pragmatique, $K = 5$, $G = (1, 35/23)$, S_2

On constate que la longueur de l'entrelaceur influence énormement les performances du codeur turbo pragmatique. On peut avoir des gains de codage de 2 dB en passant d'une longueur d'entrelaceur de 256 à 4096 pour une probabilité d'erreur de 10^{-5}. Malheureusement cela implique une augmentation du délai. En effet lorsque la taille de l'entrelaceur augmente le délai augmente également, ce qui peut ne pas être intéressant pour certaines applications en temps réel.

Pour comprendre ces résultats nous allons évaluer les bornes des performances d'erreur des codes turbo. Pour cela nous devrons calculer les distributions de poids des mots de code. Un mot de code turbo est constitué de la séquence des bits d'information, de la première séquence de parité et de la deuxième séquence de parité. Soit w le poids de la séquence d'information et z_1, z_2 les poids de la première et de la deuxième séquence de parité respectivement. Le poids du mot de code correspondant sera $d = w+z_1+z_2$ [14]. Si la distribution de poids du codeur est connue, elle pourra être utilisée pour déterminer le poids de la première séquence de parité. Cependant le poids de la seconde séquence de parité est plus complexe à déterminer

à cause de la présence de l'entrelaceur. Pour palier à ce problème Benedetto [6], [5] définit l'entrelaceur comme étant un composant probabiliste capable de produire pour une séquence d'entrée de longueur N et de poids w, l'ensemble de toutes les $\binom{N}{w}$ permutations avec une probabilité égale chacune à $1/\binom{N}{w}$. Pour un entrelaceur uniforme de taille N, il existe $N!$ permutations possibles chacune de probabilité $1/N!$. Si le poids de la séquence d'entrée est w, il y aura $w!(N-w)!$ permutations qui génereront la même séquence de sortie. Ainsi la probabilité pour chaque permutation distincte est :

$$\frac{w!(N-w)!}{N!} = \frac{1}{\binom{N}{w}} \tag{4.42}$$

Dans ce cas, on peut évaluer la distribution des poids du code turbo par :

$$A^{C}_{w,z} = \frac{A^{C_1}_{w,z_1} A^{C_2}_{w,z_2}}{\binom{N}{w}} \tag{4.43}$$

Où $A^{C_1}_{w,z_1}$ et $A^{C_2}_{w,z_2}$ sont respectivement les fonctions qui déterminent la distribution des poids pour toutes les séquences binaires de poids w du premier et du deuxième codeur avec $d = z_1 + z_2$. La borne union pour la probabilité d'erreur est donnée par :

$$P_b \leq \sum_{d=d_{free}} B_d Q\left(\sqrt{2dR\frac{E_b}{N_0}}\right) \tag{4.44}$$

Les coefficients d'erreur B_d sont donnés par l'expression :

$$\begin{aligned} B_d &= \sum_{d=w+z} \frac{w}{N} . A^{C}_{w,z} \\ &= \sum_{d=w+z_1+z_2} \frac{w}{N} . \frac{A^{C_1}_{w,z_1} . A^{C_2}_{w,z_2}}{\binom{N}{w}} \end{aligned} \tag{4.45}$$

Comme nous le remarquons la borne union est une fonction inverse de la taille de l'entrelaceur. Plus la taille de l'entrelaceur est grande, meilleures sont les performances d'erreur, ce qui est bien vérifié par les courbes 4.32 et 4.33.

4.5 Conclusion

Dans ce chapitre nous avons étudié les modulations multi-niveaux. Nous avons présenté en bref le codage multi-niveaux et le schéma TTCM. Le schéma turbo pragmatique a été étudié plus en détails. Ce dernier est particulièrement intéressant à cause de son caractère pragmatique. Nous nous sommes intéressés à l'influence que pouvait avoir l'assignation des bits à la constellation sur les performances d'erreur. Il était important de voir cette influence puisque d'une assignation à une autre on s'est retrouvé avec d'énormes gains de codage. L'influence de certains paramètres comme l'efficacité spectrale, la longueur d'entrelaceur et de contrainte a été également étudié. L'étude de ce dernier paramètre nous a revélé des résultats assez surprenants. Les longueurs de contrainte $K = 6$ et $K = 7$ n'apporte aucun gain de codage par rapport à $K = 5$. Dans le chapitre suivant nous étudierons plus en détails le schéma TTCM puis nous ferons une comparaison de ces 2 schémas de modulation.

CHAPITRE 5

ETUDE DES MODULATIONS MULTI-NIVEAUX: SCHÉMA TTCM

5.1 Introduction

Dans le chapitre précédent nous avons présenté un schéma de modulation multi-niveaux : le schéma turbo pragmatique. Avec ce schéma on a une perte d'optimalité au niveau du décodage. Pour effectuer un décodage turbo, il faut que le décodeur ait à sa disposition le bit systématique et les bits de parité bruités. Au niveau du codeur pragmatique, ces bits sont assignés à un symbole multi-niveaux qui est ensuite transmis dans le canal bruité. Le décodeur ne reçoit donc plus ces bits mais plutôt un symbole. Pour effectuer le décodage, le décodeur doit tout d'abord retrouver les bits qui ont constitués le symbole reçu. Cette fonction est assurée par un module qui calcule une mesure de fiabilité sur les bits qui ont constitué le symbole ce qui entraîne une perte d'optimalité du décodeur. Nous allons étudier un autre schéma de modulation multi-niveaux : le schéma TTCM (modulation turbo codée en treillis) en anglais (Turbo Trellis Coded Modulation), qui va utiliser un décodeur basé non plus sur une mesure de fiabilité mais qui effectuera plutôt un décodage basé directement sur les symboles. Le schéma TTCM est en fait la concaténation en parallèle de deux codeurs TCM (modulation codée en treillis) en anglais (Trellis Coded Modulation) que nous présenterons dans la section suivante.

5.2 Présentation du codeur TCM

Le codeur TCM a été introduit pour la première fois par Ungerboeck [47], [45]. Un exemple de codeur TCM de taux de codage $R = 2/3$ associé à une modulation 8-PSK est présenté à la figure 5.1. Le codeur TCM combine le codage

et la modulation contrairement aux systèmes de transmission traditionnels où le codage est séparé de la modulation. L'avantage est qu'il permet d'optimiser la distance euclidienne entre les mots de code plutôt que la distance de Hamming. Les générateurs des codes convolutionnels sont choisis pour avoir une distance de Hamming maximale. En fait pour une modulation BPSK et QPSK la distance euclidienne et la distance de Hamming sont équivalentes. Puisque les codes turbo utilisent la modulation BPSK, l'optimisation de la distance de Hamming entre les mots de code revient à optimiser la distance euclidienne. Lorsque la modulation utilisée n'est plus du BPSK ou du QPSK, l'assignation des bits codés pour la distance de Hamming ne garantit pas qu'une bonne distance euclidienne sera obtenue quelle que soit la règle d'assignation. Par conséquent les codes classiques ayant une bonne distance de Hamming peuvent ne pas être appropriés pour une modulation M-aire [11].

En 1982, Ungerboeck [45] a cherché les meilleurs codes avec une structure de treillis utilisant des ensembles de 2^{k+1} signaux pour transmettre k bits par intervalle de modulation. Par conséquent, un code est conçu pour une modulation spécifique, c'est à dire que les conceptions du codage et de la modulation, qui traditionnellement étaient deux procédés séparés ont été combinés, d'où le nom de modulation codée. L'approche de Ungerboeck est appelée modulation codée en treillis (Trellis Coded Modulation ou TCM en anglais). Le mot treillis vient du fait que ce schéma peut être décrit par un diagramme d'état similairement au diagramme treillis des codes convolutionnels à la différence que sur les branches du treillis du codeur TCM on retrouve des symboles multi-niveaux au lieu de bits. Un tel treillis est appelé treillis non binaire. Pour plus de détail sur le codeur TCM, il serait intéressant de consulter la thèse de Chan [11].

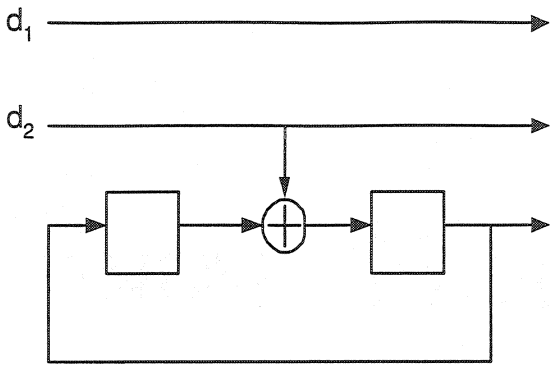

Figure 5.1: Codeur TCM à 4 états associé à la modulation 8-PSK

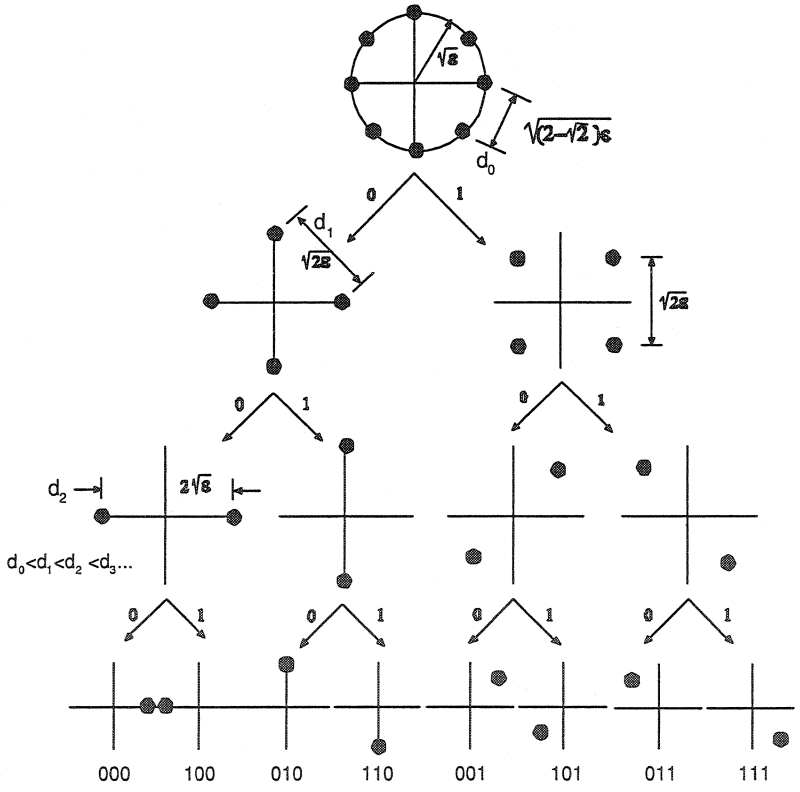

Figure 5.2: Partitionnement selon Ungerboeck pour une constellation 8-PSK

Pour augmenter la distance euclidienne entre les mots de code on utilise le principe suivant. Chaque constellation est partitionnée successivement en 2, 4, 8, ... sous ensembles. Chaque partition augmente la distance euclidienne entre les éléments d'un sous ensembles, [45]. L'illustration du partitionnement selon Ungerboeck est décrit à la figure 5.2 pour une constellation 8-PSK. Un bloc d'information D de longueur N est divisé en petits groupes de longueur k. Chaque groupe de k bits est également divisé en deux groupes de longueur n et $k-n$. Les n bits d_i, i allant de 1 à n seront codés par un codeur convolutionnel de taux de codage R égale à $n/n+1$. Les symboles codés Z_j à la sortie du codeur convolutionnel serviront à la sélection d'un sous ensemble et les $(k-n)$ bits non codés d_j, $n+1 \leq j \leq k$, à celui d'un élément du sous ensemble (voir la figure 5.3). Le taux de codage global du codeur TCM sera donc $R = k/k+1$. Si M est l'ordre de la constellation multiniveaux utilisé, $m = log_2 M$ sera le nombre de bits nécessaire à la formation d'un symbole. Le taux de codage R sera donc égale à $(m-1)/m$. Ce schéma permet d'atteindre une efficacité spectrale de k bits/s/Hz. k bits sont transmis dans un symbole d'une constellation de 2^m signaux.

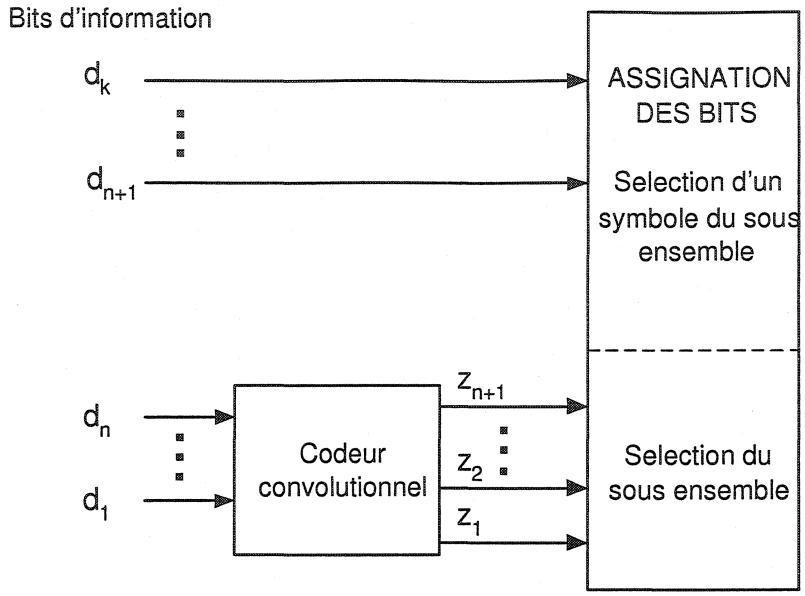

Figure 5.3: Structure du codeur/modulateur TCM de taux de codage $R = k/k+1$ avec $(k-n)$ bits non codés

Etant donné que n bits d'informations sont codés, le treillis aura 2^n branches par état sans compter les transitions parallèles. Les transitions parallèles dans le treillis se produisent lorsqu'on a un ou plusieurs bits d'informations non codés comme dans l'exemple des figures 5.4 et 5.5. La présence de branches parallèles permet d'augmenter la distance entre les chemins de longueur supérieure à une branche et d'avoir une distance libre plus grande. Les codes ayant un nombre d'états faible devraient donc avoir des transitions parallèles [46]. Cependant, avec des transitions parallèles, la distance maximale libre est limitée par la distance entre les signaux assignés aux transitions parallèles. Par conséquent les codes ayant un plus grand nombre d'états ne doivent pas avoir de transitions parallèles [11]. La distance libre du code de la figure 5.4 est donné par la distance entre deux branches parallèles, elle est égale à 4 dans l'exemple considéré. Ungerboeck a défini trois règles nécessaires pour donner les meilleurs codes TCM [45]:

1. Les transitions parallèles (quand il n'y en a) reçoivent les signaux ayant la plus grande distance euclidienne entre eux.

2. Les transitions émanant de ou convergeant vers le même état dans le treillis doivent être assignées à des sous ensembles séparés par la plus grande distance euclidienne possible.

3. Tous les signaux doivent apparaître avec une fréquence égale.

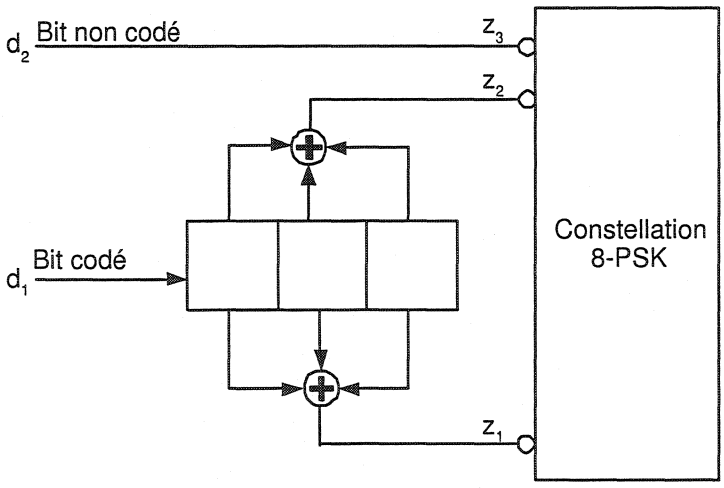

Figure 5.4: Exemple de codeur TCM, $k = 2$, $n = 1$

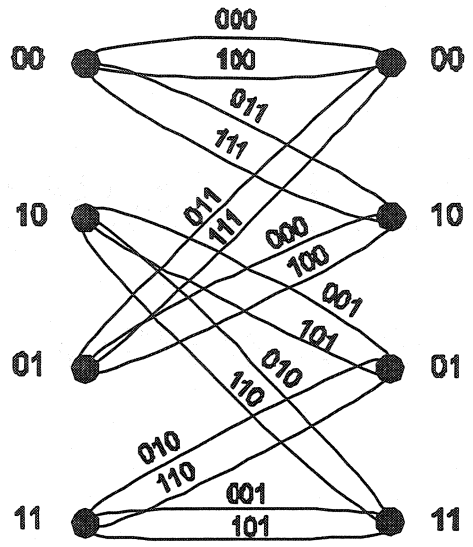

Figure 5.5: Sous treillis avec transitions parallèles correpondant au codeur de la figure 5.4

Le décodage TCM se déroule en deux étapes. La première consiste à déterminer le symbole le plus rapproché en terme de distance euclidienne du symbole reçu. Dans la seconde étape on effectue un décodage de Viterbi ayant pour métriques les distances euclidiennes calculées précédemment.

5.3 Présentation du schéma TTCM (Turbo Treillis Coded Modulation)

Le schéma TTCM a été présenté par Robertson [41]. Sa structure générale est présentée dans la figure 4.5. La figure 5.6 quant à elle est un exemple d'un codeur TTCM 8-PSK à 8 états.

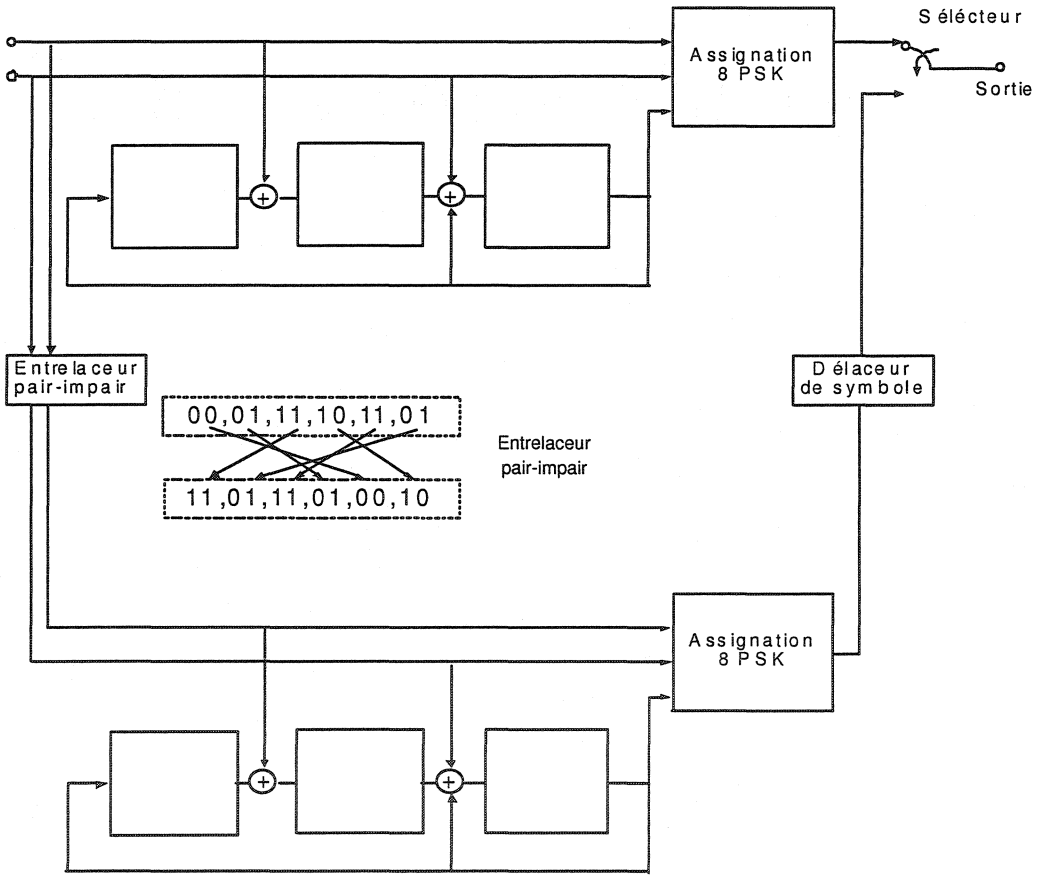

Figure 5.6: Codeur TTCM 8-PSK à 8 états

Au niveau du schéma TTCM on applique le principe turbo au codeur TCM, l'idée étant de retenir les avantages des deux structures: les performances d'erreurs remarquables des codes turbo et les avantages en bande passante des TCM. Le schéma TTCM a une structure semblable à celle des codes turbo à la différence que

des codeurs TCM sont utilisés au lieu des codeurs convolutionnels. On note cependant des différences majeures à savoir : l'entrelaceur opère sur des groupes de m bits plutôt que sur un bit et le désentrelaceur opère sur des symboles. L'entrelaceur doit en outre être obligatoirement de type pair-impair [41] pour ne pas qu'on ait à perforer un symbole d'information ou à transmettre deux fois le même symbole à cause de la présence du sélecteur à la sortie du codeur TTCM. Cette contrainte doit être respectée aussi parce qu'au niveau du décodage chaque décodeur reçoit alternativement un symbole du codeur TCM lui correspondant et un autre de l'autre codeur TCM. On effectue une perforation du symbole ne le concernant pas et cette perforation s'effectue selon la parité de l'ordre du symbole reçu. Les symboles doivent donc garder leur ordre d'arrivée pour éviter que les décodeurs ne travaillent avec le symbole qui ne le concerne pas ou que ce symbole soit perforé. Il faut noter qu'une autre approche de schéma multi-niveaux utilisant des codeurs TCM en parallèle a été proposé par Benedetto, Divsalar, Montorsi et Pollara [3], [4], où il n' y a pas de perforation de bits codés ou de symboles codés. La structure du schéma qu'ils ont proposé est montré dans la figure 5.7.

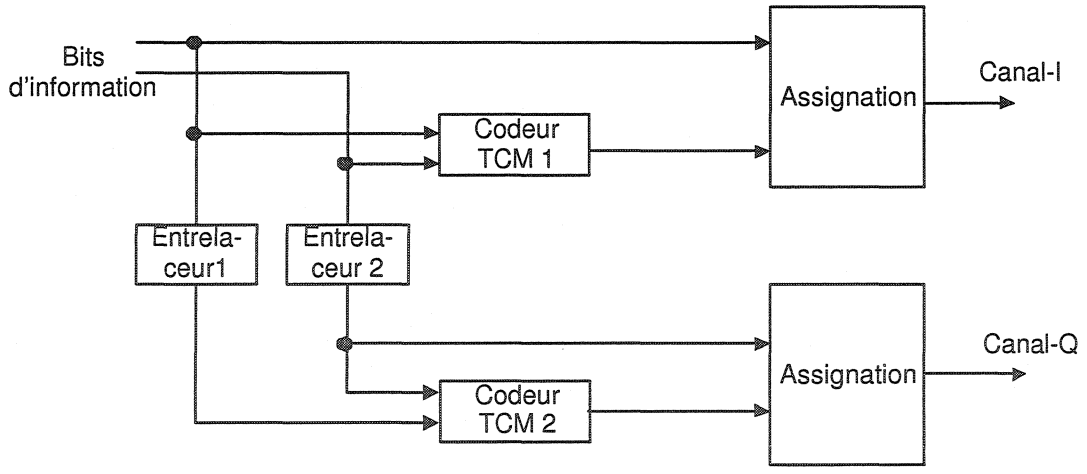

Figure 5.7: Codeur TTCM de [3] et [4]

Ici, à partir du symbole de parité d'un codeur TCM et d'un bit systématique, on prend la forme entrelacée pour le deuxième codeur et on forme un symbole de modulation. Ces symboles sont ensuite envoyés tel que décrit dans la figure 5.7 en utilisant une modulation en phase et en quadrature pour former un signal composite. Les bits de parité et systématiques peuvent être également assignés à une constellation de la forme 8-PSK*8-PSK. Dans ce cas les symboles 8-PSK seront envoyés en serie.

5.3.1 Transitions parallèles

Les travaux de Ungerboeck [46] ont montré que les meilleurs codeurs TCM ayant un faible nombre d'état possèdent des transitions parallèles. Il serait donc intéressant d'utiliser des codeurs TCM ayant des transitions parallèles dans les schémas 5.6 et 5.7. Malheureusement ces schémas ne doivent pas avoir de transitions parallèles [41] afin que chaque bit d'information puisse bénéficier de l'entrelacement. Cette condition peut cependant être relaxée sous deux conditions: la première est lorsqu'on désire une grande efficacité spectrale [41], on travaille avec des rapports signal sur bruit importants et donc la distance euclidienne qui sépare les signaux des sous ensembles est beaucoup plus grande. La seconde condition sous laquelle on permet des transitions parallèles ([10] et [9]) est que l'entrelaceur ne garde pas chaque groupe de k bits d'informations inchangés. Pour que cette condition soit toujours respectée, Blackert [10] a modifié le schéma TTCM [41] proposé par Robertson. En fait il s'agit de la même structure sauf qu'après l'entrelacement des bits on effectue un mapping. Ce schéma est presenté dans la figure 5.8.

Il y a dans ce cas deux modifications qu'il faut apporter au décodeur pour qu'il décode correctement le codeur TTCM utilisant des transitions parallèles. La première est qu'à la sortie du décodeur, les bits décodés ne doivent pas seulement être désentrelacés mais ils doivent être également "dé-assignés". La seconde modification doit être faite au niveau du bloc de calcul de métriques. Ce dernier devra tenir compte du fait le second codeur TCM a travaillé sur une version "mappée" des bits d'informations.

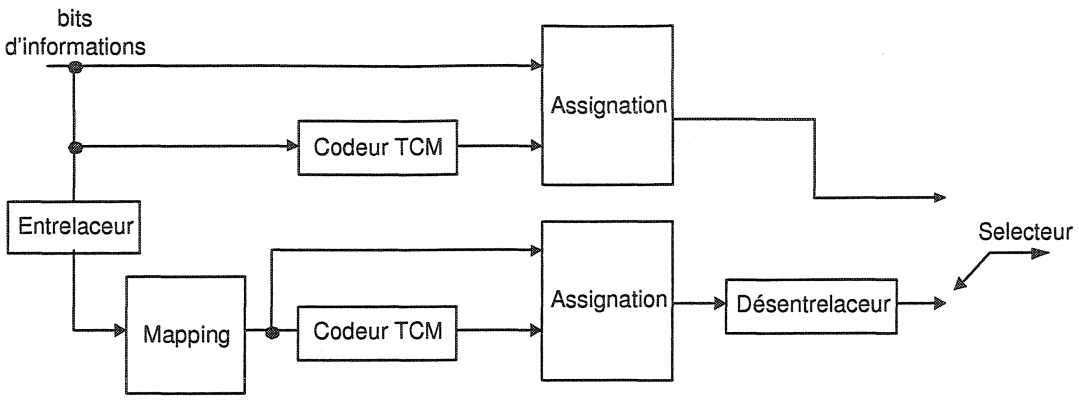

Figure 5.8: Codeur TTCM autorisant les transitions parallèles

5.4 Le décodage TTCM

Le décodeur TTCM [41] est basé sur le même principe que le décodeur turbo binaire. Le principe du décodeur TTCM est présenté à la figure 5.9. On effectue un décodage itératif à la différence que l'information passée d'un décodeur à un autre n'est pas la même. Un autre point important est que chaque décodeur voit alternativement un symbole de sortie lui correspondant et un autre correspondant à l'autre décodeur. Les bits d'information c'est à dire systématiques qui ont servi à l'assignation de ces deux symboles ne posent pas de problème parce qu'ils sont identiques et donc les décodeurs recevront les mêmes bits d'information pour les deux symboles de sortie reçus. Ce n'est par contre pas le cas en ce qui concerne les bits de parité. Lors du décodage d'un groupe de k bits d'information (k étant l'efficacité spectrale du schéma), les décodeur 1 et 2 recevront le bit de parité du codeur 1 ainsi que celui du codeur 2, ce qui est contraire au principe du décodage itératif du codeur turbo binaire. Nous avons indexé par une étoile "*" les symboles qui doivent être ignorés lors du décodage, ils seront tout simplement perforés et la métrique de branche lui correspondant sera nulle.

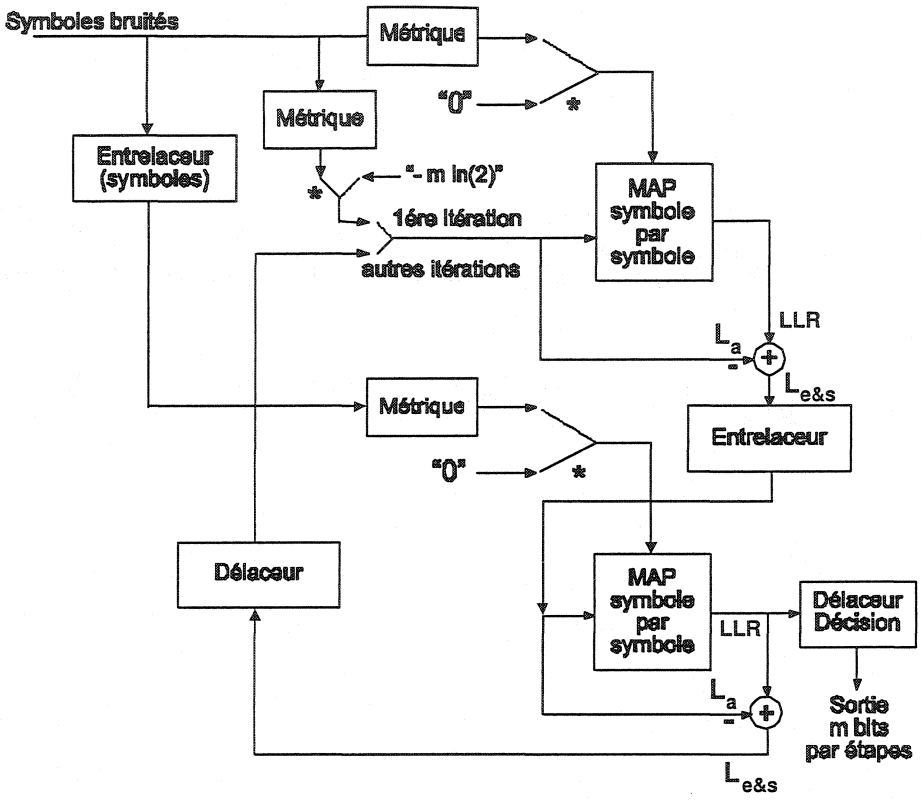

Figure 5.9: Principe du décodeur TTCM

Au niveau de l'algorithme de décodage il ne sera plus possible d'utiliser l'algorithme MAP et LogMAP décrit dans le chapitre 2 étant donné qu'on travaille ici sur des treillis non binaires. Il va donc falloir modifier cet algorithme afin de l'adapter à des symboles non binaires. Soit 2^A l'ordre de la modulation utilisée, on disposera donc de 2^A symboles dans la constellation de signaux. On devra déterminer à l'instant k le symbole i qui est le plus probable. Pour cela on calcule les 2^A probabilités, $P(d_k = i|\mathbf{R}_1^N)$ pour i allant de 0 à $2^A - 1$ sachant qu'on a reçu le signal $\mathbf{R}_1^N$. Le symbole retenu sera celui dont la probabilité d'erreur est la plus élévée. Dans le cas d'un treillis non binaire on devra effectuer 2^A rapport de vraisemblance $LLR(d_{k,i})$, un seul rapport n'est plus suffisant car on dispose de plus de deux symboles. La décision se fera suivant le rapport optimum, le symbole décodé sera celui dont la rapport de vraisemblance aura la valeur minimale. On rappelle que le LLR est défini comme selon l'équation (5.1) :

$$LLR(d_{k,i}) = ln\frac{P(d_k = i|\mathbf{R}_1^N)}{P(d_k = 0|\mathbf{R}_1^N)} \tag{5.1}$$

Le développement du MAP [41] est similaire à celui que nous avons présenté dans le chapitre 2:

$$P(d_k = i|\mathbf{R}_1^N) = \sum_{m=0}^{2^{K-1}-1} \alpha_k^i(m)\beta_k^i(m) \quad , \; i \in \{0, ..., 2^A - 1\} \tag{5.2}$$

Avec les métriques avant, arrière et de branche:

$$\alpha_k^i(m) = \delta_k^i(m) \sum_{j=0}^{2^A-1} \alpha_{k-1}^j(S_B^j(m)) \tag{5.3}$$

$$\beta_k^i(m) = \sum_{j=0}^{2^A-1} \beta_{k+1}^j(S_F^i(m))\delta_{k+1}^j(S_F^i(m))) \tag{5.4}$$

$$\delta_k^i(m) = \frac{1}{2^A} P(\mathbf{R}_k|d_k = i, S_k = m) \tag{5.5}$$

La métrique de branche doit désormais s'exprimer en fonction des symboles transmis dans le canal. Soit $\mathbf{X}_k$ le symbole émis à l'instant k et $\mathbf{R}_k$ le symbole reçu correspondant. Nous avons alors :

$$\delta_k^i(m) = \frac{1}{2^A}\frac{1}{2\pi\sigma^2} exp\Big(- \frac{(\mathbf{R}_k - \mathbf{X}_k(i,m))^2}{2\sigma^2}\Big) \tag{5.6}$$

Dans le cas du turbo binaire, on a un coefficient $1/2$ dans l'expression de la métrique de branche qui vient du fait qu'on a seulement deux symboles 0 et 1 et que ces deux symboles sont équiprobables. Dans le cas du décodeur MAP symbole par symbole, étant donné qu'on a 2^A symboles, ce coefficient devient $1/2^A$ en supposant toujours que les symboles sont équiprobables. En passant dans le domaine logarithmique, les métriques en avant $A_k^i(m)$, en arrière $B_k^i(m)$ et d'état $D_k^i(m)$ deviennent :

$$A_k^i(m) = D_k^i(m) + \boldsymbol{E}_{j=0}^{2^A-1}\Big(A_{k-1}^j\big(S_B^j(m)\big)\Big) \tag{5.7}$$

$$B_k^i(m) = \boldsymbol{E}_{j=0}^{2^A-1}\Big(B_{k+1}^j\big(S_F(m)\big) + D_{k+1}^j\big(S_F(m)\big)\Big) \tag{5.8}$$

$$D_k^i(m) = \frac{1}{2\sigma^2}\big(\mathbf{R}_k - \mathbf{X}_k(i,m)\big)^2 \tag{5.9}$$

On rappelle que la fonction $\mathbf{E}$ est défini comme suit, pour plus de détails voir (3.30), (3.31) et (3.32):

$$\mathbf{E}_n(x(n)) = ln(\sum_n e^{x(n)}) \tag{5.10}$$

Au niveau du décodage turbo binaire nous avons vu au chapitre 2 et plus précisement à la section 3.5.1.3, que le $LLR(d_k)$ à la sortie d'un décodeur turbo peut se décomposer en trois éléments qui sont l'information extrinsèque Le_k, l'information a priori La_k et l'information systématique Li_k. L'information systématique provient des bits systématiques, l'information a priori provient du décodeur précédent et l'information extrinsèque dépend de toutes les autres entrées et c'est uniquement cette dernière qui est transmise d'un décodeur à un autre, (voir la figure 3.18). Dans le cas du décodeur TTCM, la situation est plus complexe parce que l'information extrinsèque ne peut être séparée de l'information systématique étant donné que cette dernière est associée à l'information de parité pour former un symbole. C'est le même bruit qui affecte les deux informations contrairement au codage turbo binaire où le bit de parité est transmis séparement du bit systématique et donc les bruits affectant ces deux signaux sont indépendants. Le LLR à la sortie de chaque décodeur TTCM sera divisé en deux éléments à savoir l'information a priori La_k et l'information extrinsèque et systématique notée: $Le\&s_k$ et ce n'est que cette dernière qui sera transmise au décodeur suivant. Son expression est donnée par (5.11).

$$L_{e\&s}(d_k = i) = ln(P(d_k = i|\mathbf{R}_1^N)) - ln(P(d_k = i)) \tag{5.11}$$

L'information a priori quant à elle est donnée par l'équation suivante :

$$L_a(d_k = i) = ln(P(d_k = i)) \tag{5.12}$$

(5.11) devient donc :

$$L_{e\&s}(d_k = i) = ln(P(d_k = i|\mathbf{R}_1^N)) - L_a(d_k = i) \tag{5.13}$$

5.5 Paramètres de simulations

Toutes les simulations qui sont présentées dans ce chapitre ont été faites pour un bruit gaussien. Nous avons utilisé essentiellement les longueurs d'entrelaceurs N=256, 900, 2052 et 4158 pour effectuer les simulations. Pour les longueurs de contrainte du codeur, nous avons surtout utilisé $K = 4$ parce qu'elle nous permet d'avoir un bon compromis entre la complexité et la performance d'erreur. Plus de détails sont donnés dans la section 5.6. Les polynômes générateurs des codes optimaux c'est à dire ceux qui maximisent la distance libre du codeur et qui ont été utilisés pour les simulations sont recencés dans le tableau 5.1 [46], [41].

Les vecteurs générateurs des codes TCM sont donnés par les coefficients H^i, i allant de 0 à $m - 1$. m représente le nombre de bits nécessaire pour former un symbole. Par exemple pour une modulation 16-QAM, m est égal à 4. On rappelle que dans le codeur TCM nous avons $m - 1$ bits d'information et un bit de parité. A chaque bit i formant un symbole on associe un générateur, H^i. H^0 est le générateur associé au premier symbole d'information, H^i celui associé au $(i + 1)$ symbole d'information et H^{m-1}, le générateur associé au bit de parité. $H^i = (h_0^i \dots h_j^i \dots h_{K-1}^i)$, h_j^i est égal à 1 s'il y a une connection entre la $k^{\text{ième}}$ cellule du registre à décalage et le bit i. Sinon, il est égal à 0. Les générateurs H^i sont exprimés en octal.

Tableau 5.1: Vecteurs générateurs des codeurs TCM avec la meilleure distance minimale pour la modulation 8-PSK et 16-QAM [46], [41]

Modulation	K	m	$H^0(D)$	$H^1(D)$	$H^2(D)$	$H^3(D)$
8-PSK	4	2	04	02	11	-
8-PSK	5	2	16	04	23	-
8-PSK	6	2	34	16	45	-
8-PSK	7	2	074	036	105	-
16-PSK, 16-QAM	4	3	10	04	02	11
16-PSK, 16-QAM	5	3	10	04	02	21
16-PSK, 16-QAM	6	3	10	04	02	41
16-PSK, 16-QAM	7	3	044	014	002	101

Les simulations ont été réalisées pour les modulations 8-PSK et 16-QAM, soit pour une efficacité spectrale de 2 et 3 bits/s/Hz avec un taux de codage respectif de 2/3 et 3/4. On rappelle que le taux de codage doit être de la forme $k/k+1$. Pour ne pas effectuer trop d'effort de calcul sans grand gain de codage on a effectué des simulations pour les deux constellations étudiées, le 8-PSK et le 16-QAM afin de voir le nombre d'itérations qui serait le plus intéressant. Les résultats de ces simulations sont présentés aux figures 5.10 et 5.11. Pour les simulations qui suivent nous montrons seulement les résultats de l'itération 6 car elle nous donne un bon compromis entre la complexité et la performance d'erreur pour toutes les constellations utilisées. Au delà de l'itération 6 on n'obtient pratiquement plus de gain de codage et donc nous nous limiterons à l'itération 6.

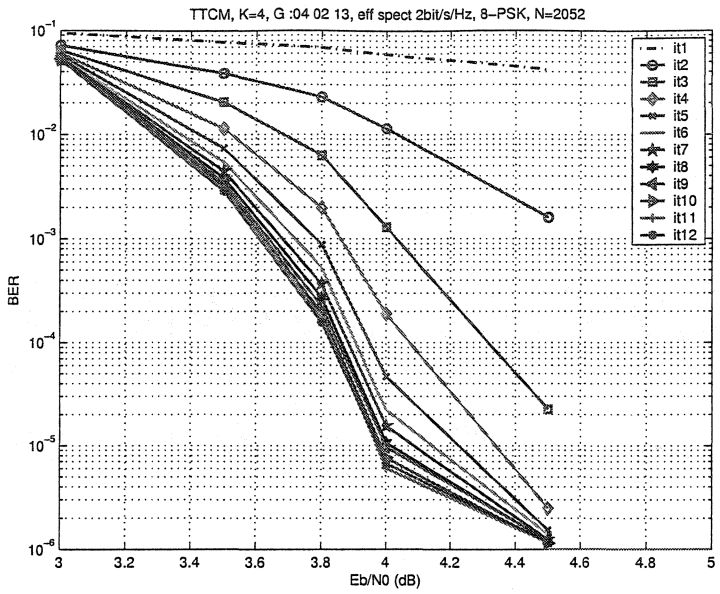

Figure 5.10: Influence du nombre d'itération sur les performances du décodeur TTCM, 8-PSK, $N = 2052$, $G = (040213)$

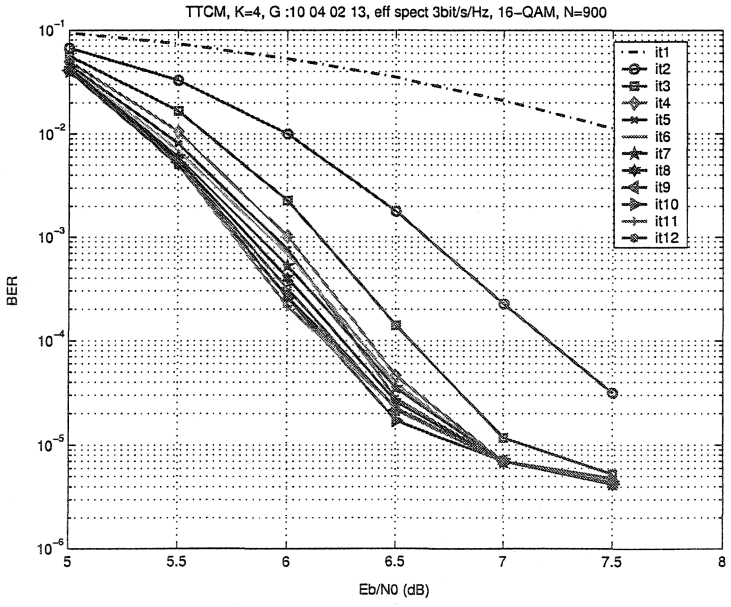

Figure 5.11: Influence du nombre d'itération sur les performances du décodeur TTCM, 16-QAM, $N = 900$, $G = (040213)$

Nous avons vu dans la section 4.2 que l'assignation des bits à la constellation se fait selon l'assignation de Ungerboeck plutôt que selon l'assignation de Gray. Comme l'assignation de Ungerboeck permet de maximiser la distance euclidienne entre les symboles, ses performances d'erreur devraient être meilleures. Nous avons donc effectué des simulations pour différentes longueurs de contrainte, d'entrelaceur et d'ordre de modulation pour fins de vérification. Les résultats obtenus sont recensés dans les courbes 5.12 à 5.15. On obtient des gains de codage allant jusqu'à 4 dB d'un partionnement à un autre pour une longueur d'entrelaceur N=4158 et une efficacité spectrale de 3 bits/s/Hz. On remarque également que le gain de codage dû au partitionnement de Ungerboeck augmente avec la longueur de l'entrelaceur.

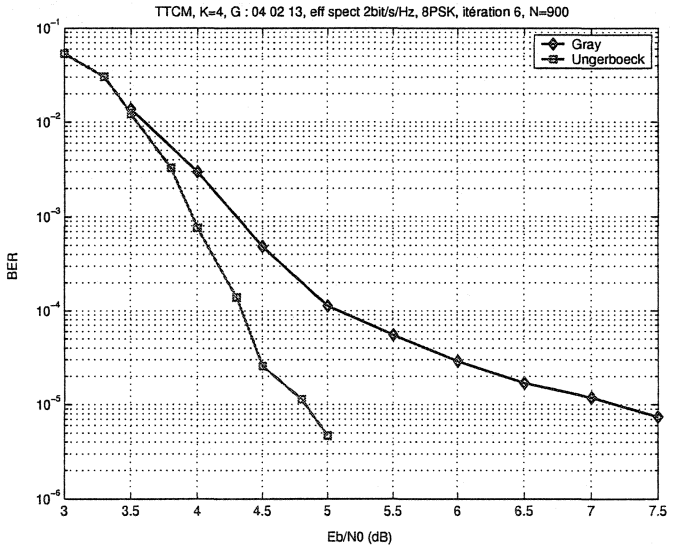

Figure 5.12: Comparaison de l'assignation selon Ungerboeck et selon Gray des bits à la constellation sur les performances du schéma TTCM, 8-PSK, $N = 900$, $K = 4$,$G : 040213$.

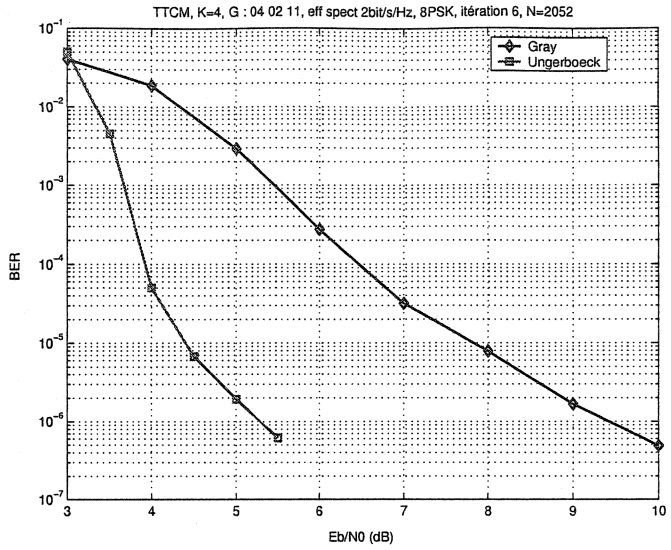

Figure 5.13: Comparaison de l'assignation selon Ungerboeck et selon Gray des bits à la constellation sur les performances du schéma TTCM, 8-PSK, $N = 2052$, $K = 4$, $G : 040211$

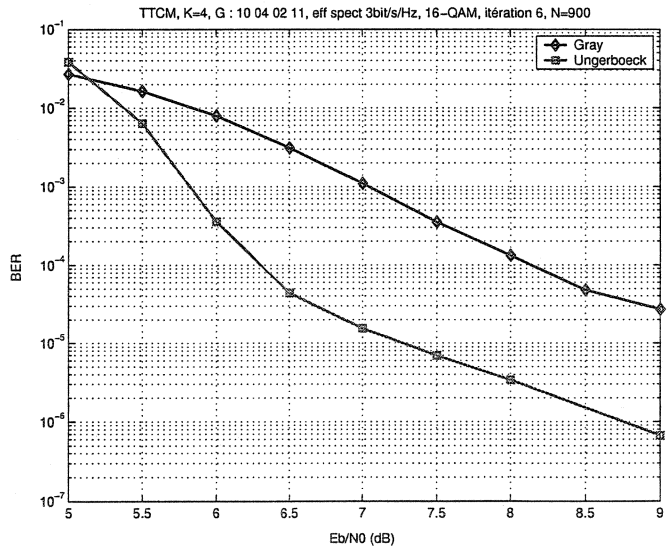

Figure 5.14: Comparaison de l'assignation selon Ungerboeck et selon Gray des bits à la constellation sur les performances du schéma TTCM, 16-QAM, $N = 900$, $K = 4$, $G : 10040211$

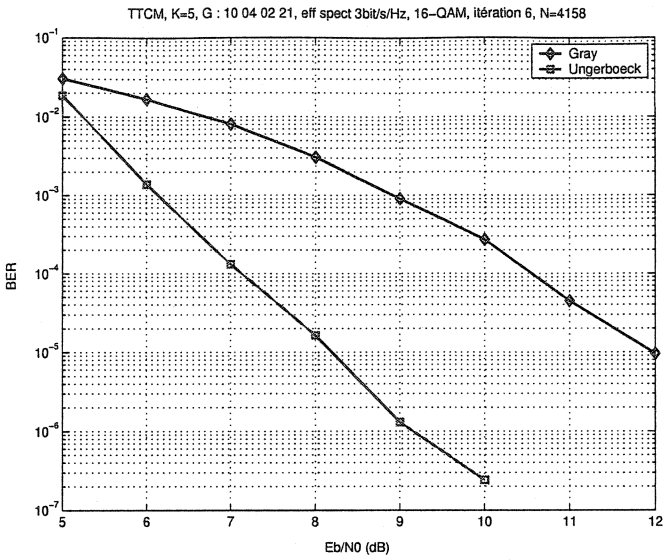

Figure 5.15: Comparaison de l'assignation selon Ungerboeck et selon Gray des bits à la constellation sur les performances du schéma TTCM, 16-QAM, $N = 4158$, $K = 5$, $G : 10040221$

5.6 Influence de la longueur de contrainte

Au niveau du schéma turbo pragmatique nous avons remarqué que l'augmentation de la longueur de contrainte n'apportait pratiquement pas d'amélioration de la probabilité d'erreur au delà de $K = 4$. Nous avons également effectué les mêmes simulations pour le schéma TTCM afin de voir l'influence de la longueur de contrainte sur les performances d'erreur du schéma TTCM et voir si on arrivait aux mêmes résultats. Les simulations ont été faites pour plusieurs longueurs d'entrelaceurs, pour le 8-PSK et le 16-QAM afin de nous assurer qu'on obtenait toujours les mêmes caractéristiques quelque soit les paramêtres utilisés. Ces résultats sont présentés dans les figures 5.16 à 5.19.

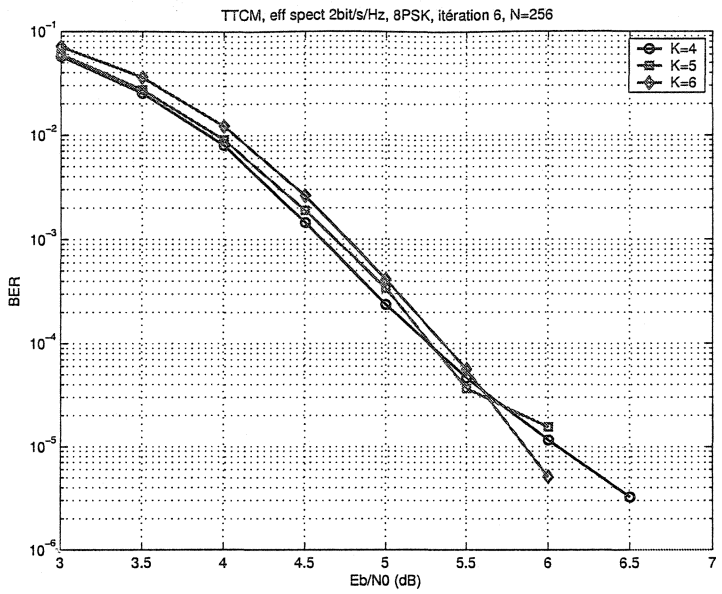

Figure 5.16: Influence de la longueur de contrainte sur les performances du schéma TTCM, 8-PSK, $N = 256$

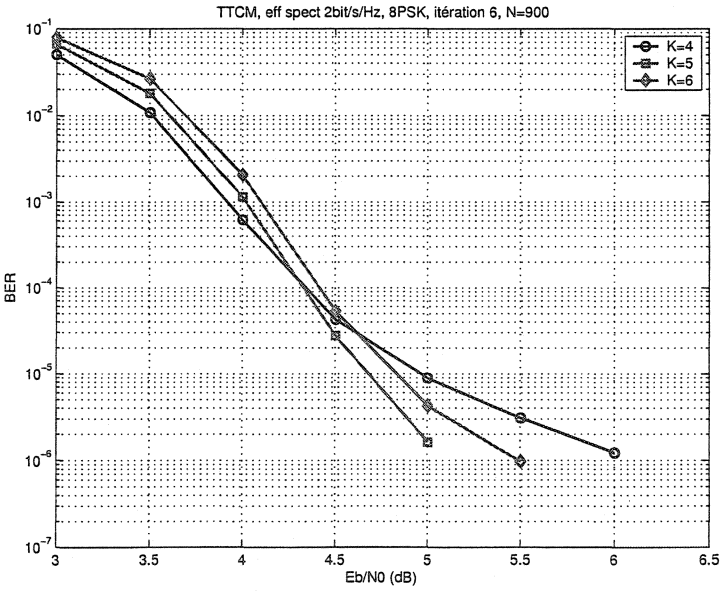

Figure 5.17: Influence de la longueur de contrainte sur les performances du schéma TTCM, 8-PSK, $N = 900$

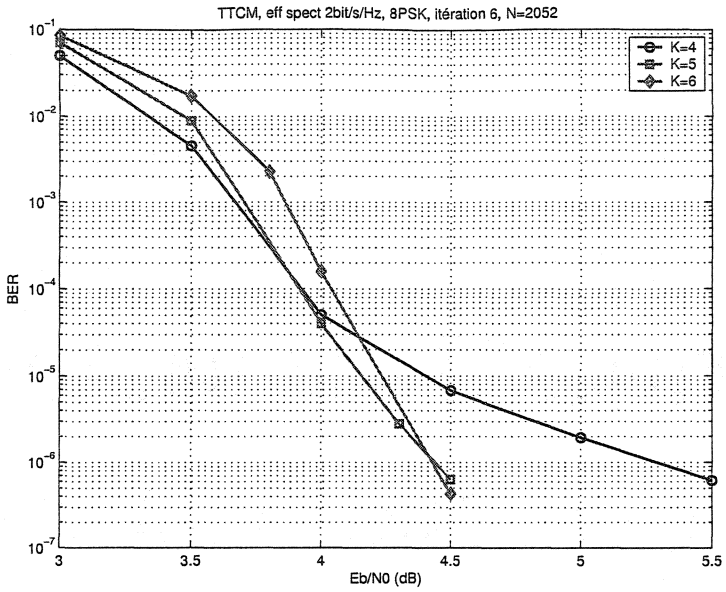

Figure 5.18: Influence de la longueur de contrainte sur les performances du schéma TTCM, 8-PSK, $N = 2052$

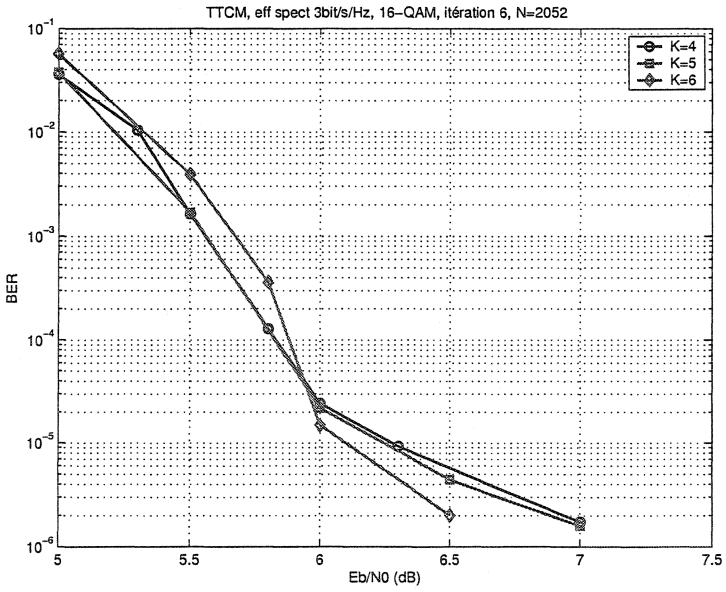

Figure 5.19: Influence de la longueur de contrainte sur les performances du schéma TTCM, 16-QAM, $N = 2052$

Nous remarquons que nous obtenons pratiquement les mêmes performances d'erreur pour $K = 4$, 5 et 6 et ce pour des longueurs d'entrelaceurs différentes . L'augmentation de la longueur de contrainte n'entraîne pratiquement aucune amélioration de la probabilité d'erreur comme nous l'avons constaté dans le cas du schéma turbo pragmatique. Nous constatons tout de même un léger gain de codage pour des probabilités d'erreurs inférieures à 10^{-4}. Ce gain s'agrandit lorsque la probabilité d'erreur diminue. Pour une probabilité d'erreur de 10^{-5}, nous obtenons un gain de codage de 0.3dB en passant de $K = 4$ à $K = 5$ pour des efficacités spectrales de 2 et 3 bits/s/Hz. Lorsque la probabilité d'erreur est de 10^{-6} le gain de codage obtenu en passant de $K = 4$ à $K = 5$ est de 0.7 dB. Les longueurs de contrainte $K = 5$ et $K = 6$ donnent les mêmes résultats à 0.1 dB près. La longueur de contrainte la plus intéressante est donc $K = 4$, au delà de laquelle nous avons une complexité supplémentaire sans toutefois avoir un gain de codage significatif. Ces résultats s'expliqueraient par le fait que nous avons employé la même structure, la structure turbo pour réaliser le schéma TTCM et donc les raisons évoquées pour le schéma turbo pragmatique le sont également pour le schéma TTCM. D'autre part il faut noter que les codes TCM utilisés pour concevoir le schéma TTCM ne donnent pas non plus une grande différence de la distance libre entre les longueurs de contrainte. Par exemple pour la modulation 8-PSK, pour $K = 4$, 5 et 6 nous avons d^2_{free}/d^2_1 égal respectivement à 2.293, 2.586 et 2.879 respectivement, [45]. En ce qui concerne la modulation 16-QAM, pour $K = 4$, 5 et 6 nous avons d^2_{free}/d^2_0 égal à 4 [46], d_0 étant la distance entre les signaux de la constellation sans partitionnement et d_1 étant la distance entre les signaux de la constellation après le premier partitionnement de Ungerboeck, voir la figure 5.2. Pour les trois longueurs de contrainte on a la même distance libre. Il est donc normal que la structure turbo du TTCM combinée aux codes TCM fasse qu'il n'y ait pratiquement aucune différence entre les performances d'erreur des trois longueurs de contrainte.

5.7 Influence de l'efficacité spectrale

Nous allons nous intéresser dans cette section à l'influence de l'efficacité spectrale sur les performances du schéma TTCM. Nous avons effectué des simulations pour N=900, 2052 et 4158 et pour $K = 4$ et $K = 5$, les résultats de ces simulations sont recensés dans les courbes 5.20 à 5.23. Nous constatons que le passage de l'efficacité spectrale 2 à 3 bits/s/Hz entraîne un rapport signal sur bruit supplémentaire d'environ 2 dB pour toutes les trois longueurs d'entrelaceur.

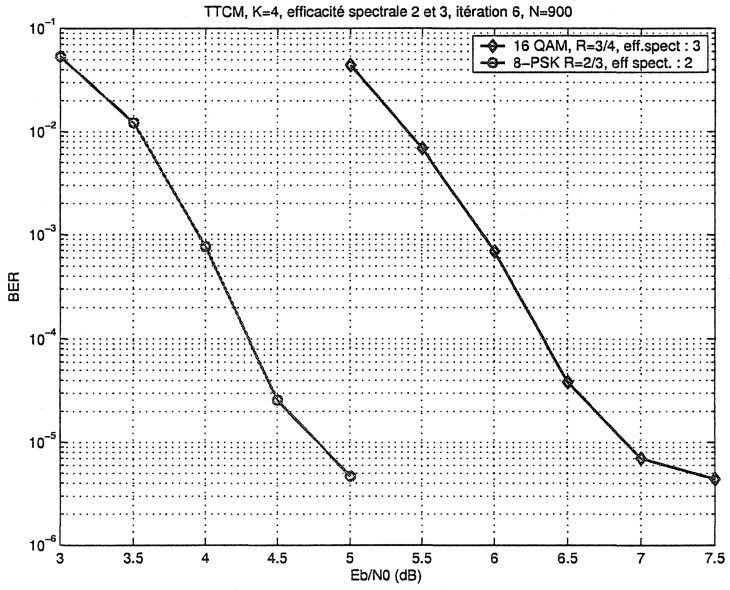

Figure 5.20: Performances du schéma TTCM pour différentes efficacités spectrales, $K = 4$, $N = 900$

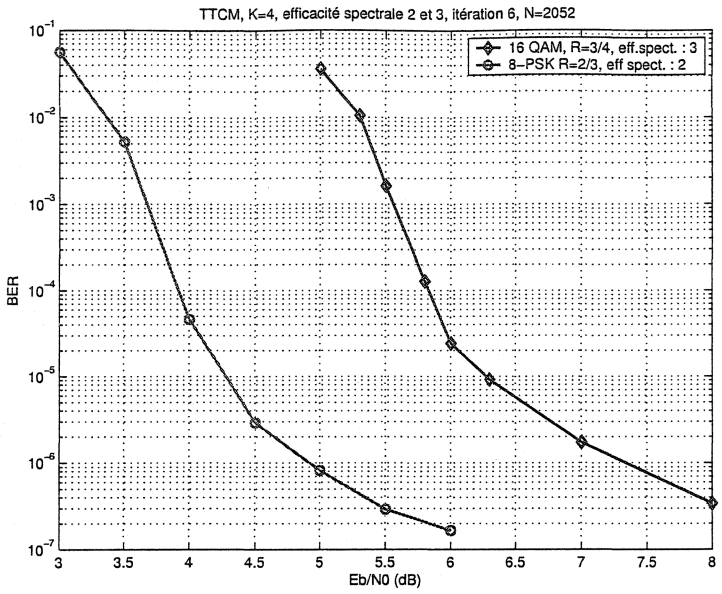

Figure 5.21: Performances du schéma TTCM pour différentes efficacités spectrales, $K = 4$, $N = 2052$

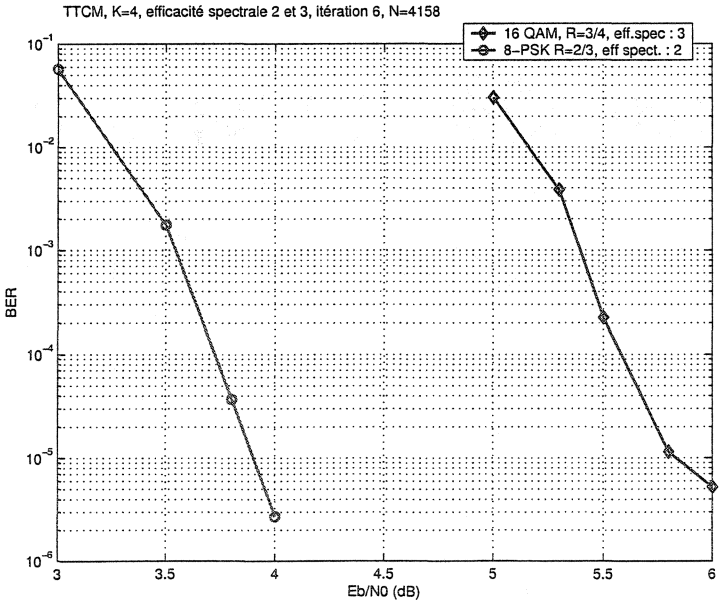

Figure 5.22: Performances du schéma TTCM pour différentes efficacités spectrales, $K = 4$, $N = 4158$

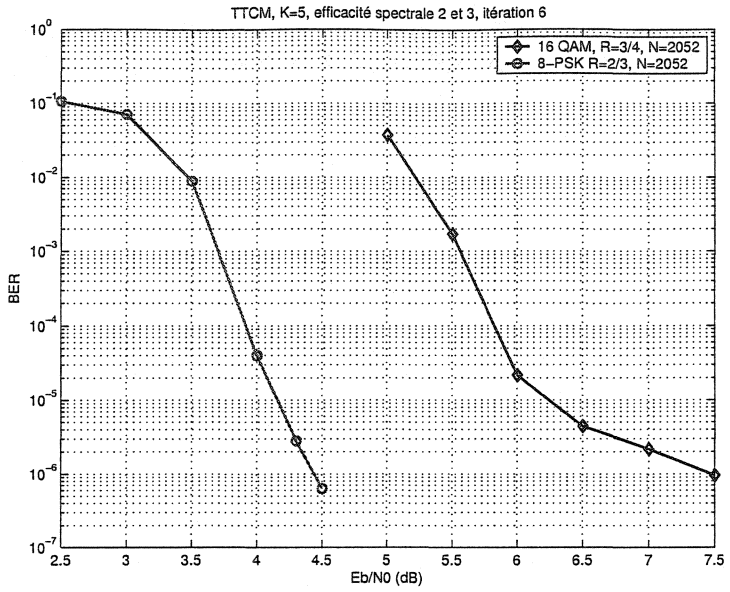

Figure 5.23: Performances du schéma TTCM pour différentes efficacités spectrales, $K = 5$, $N = 2052$

Le tableau 5.2 recense l'écart entre la limite de Shannon et la limite trouvée par les simulations du codeur TTCM pour N=900, 2052 et 4158. Pour la modulation 16-PSK par contre on obtient un très grand écart de 6 dB. Ces résultats s'expliquent parce que la distance euclidienne entre les signaux devient plus petite au fur et à mesure que le nombre de signaux de la constellation PSK augmente. Au delà d'une efficacité spectrale de 2 bits/s/Hz, il n'est plus intéressant d'utiliser la modulation PSK. Il serait préférable d'utiliser la modulation QAM bien que celle-ci soit plus complexe à mettre en oeuvre mais la distance euclidienne entre les signaux de sa constellation ne change pas.

Tableau 5.2: Ecart entre la limite théorique de Shannon et la limite pratique du codeur TTCM pour $K = 4$

Taux de codage R	N	Ordre de la modulation	Limite de Shannon (dB)	E_b/No pour BER $= 10^{-5}$ (dB)	Ecart du TTCM /à la limite de Shannon (dB)
2/3	900	8-PSK	1.76	4.8	3.04
3/4	900	16-QAM	3.67	6.9	3.23
2/3	2052	8-PSK	1.76	4.3	2.54
3/4	2052	16-PSK	3.67	9.9	6.23
3/4	2052	16-QAM	3.67	6.3	2.63
2/3	4158	8-PSK	1.76	3.8	2.04
3/4	4158	16-QAM	3.67	5.8	2.13

5.7.1 Gain de codage obtenu par le codage TTCM par rapport à la modulation non codée pour différentes efficacités spectrales

Nous avons effectué des simulations afin d'évaluer les gains de codage obtenus par rapport au système non codé qui nous aurait permis d'avoir la même efficacité spectrale. Ces simulations ont été faites pour les efficacités spectrales 2 et 3 bits/s/Hz, pour 5 longueurs d'entrelaceur : N=900, 2052, 4158, 10000 et 15000 et pour les longueurs de contrainte $K = 4$ et 5. Les courbes résultantes sont présentées aux figures 5.24 à 5.31.

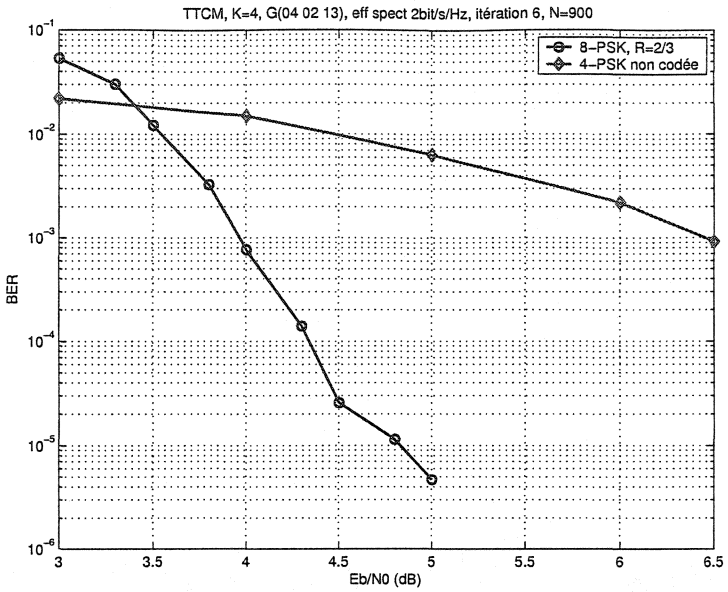

Figure 5.24: Gain de codage du schéma TTCM par rapport à la modulation non codée, efficacité spectrale 2 bits/s/Hz, $K = 4$, $N = 900$

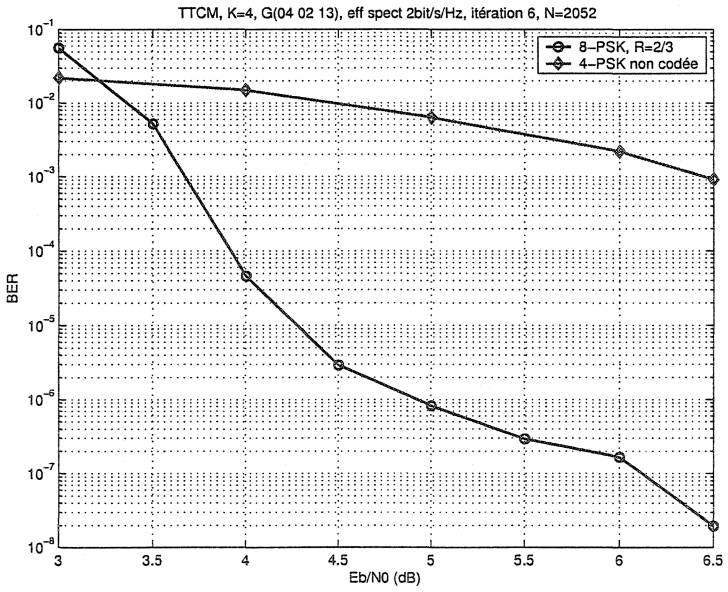

Figure 5.25: Gain de codage du schéma TTCM par rapport à la modulation non codée, efficacité spectrale 2 bits/s/Hz, $K = 4$, $N = 2052$

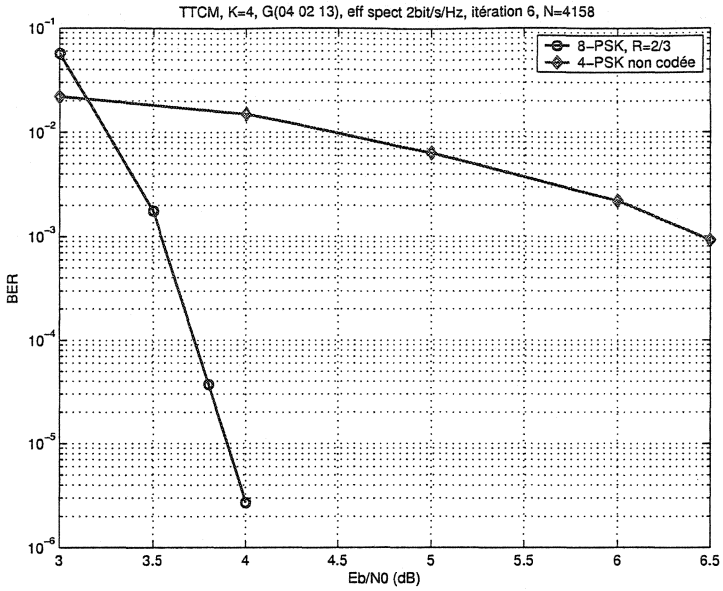

Figure 5.26: Gain de codage du schéma TTCM par rapport à la modulation non codée, efficacité spectrale 2 bits/s/Hz, $K = 4$, $N = 4158$

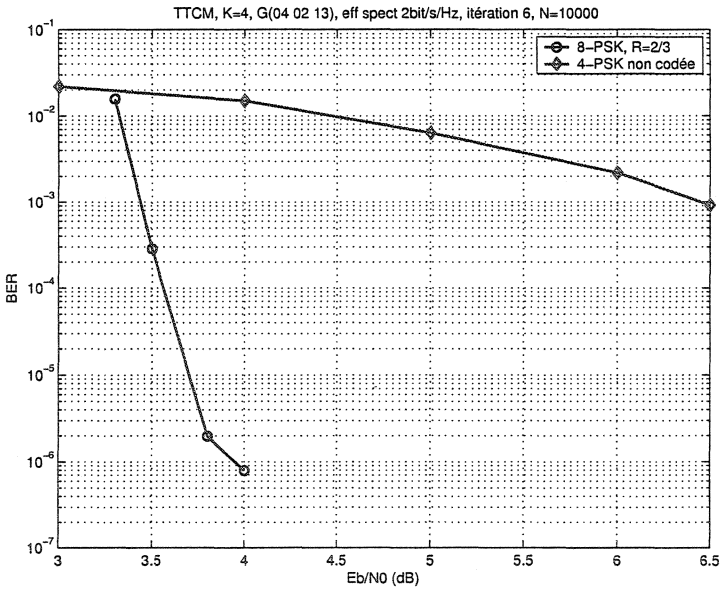

Figure 5.27: Gain de codage du schéma TTCM par rapport à la modulation non codée, efficacité spectrale 2 bits/s/Hz, $K = 4$, $N = 10000$

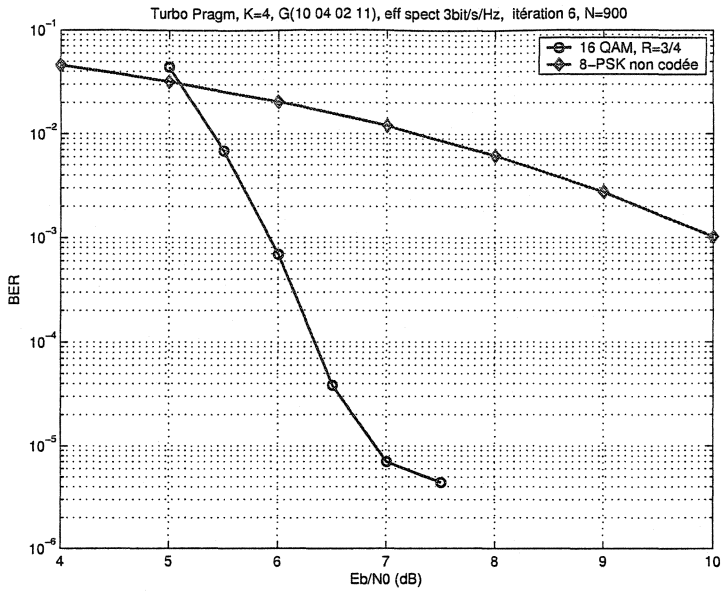

Figure 5.28: Gain de codage du schéma TTCM par rapport à la modulation non codée, efficacité spectrale 3 bits/s/Hz, $K = 4$, $N = 900$

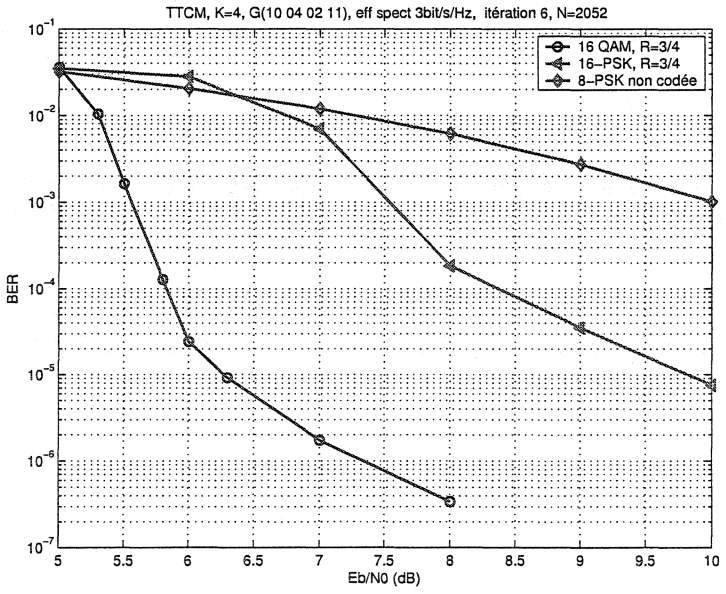

Figure 5.29: Gain de codage du schéma TTCM par rapport à la modulation non codée, efficacité spectrale 3 bits/s/Hz, $K = 4$, $N = 2052$

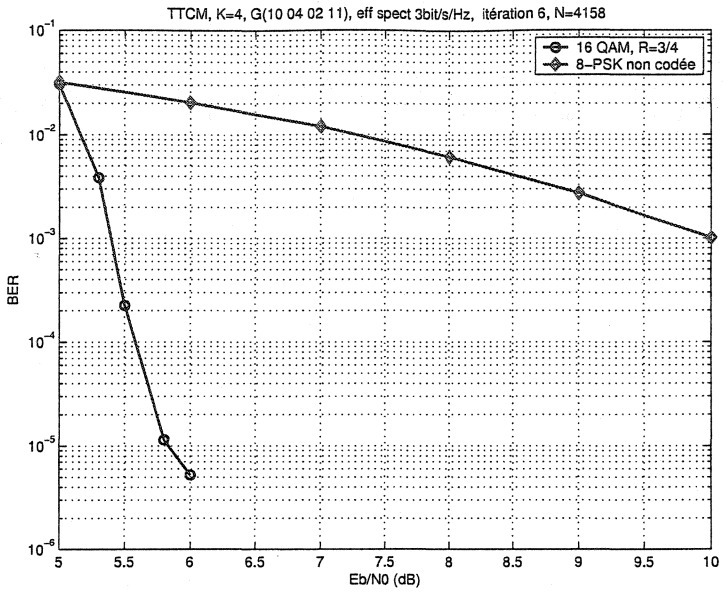

Figure 5.30: Gain de codage du schéma TTCM par rapport à la modulation non codée, efficacité spectrale 3 bits/s/Hz, $K = 4$, $N = 4158$

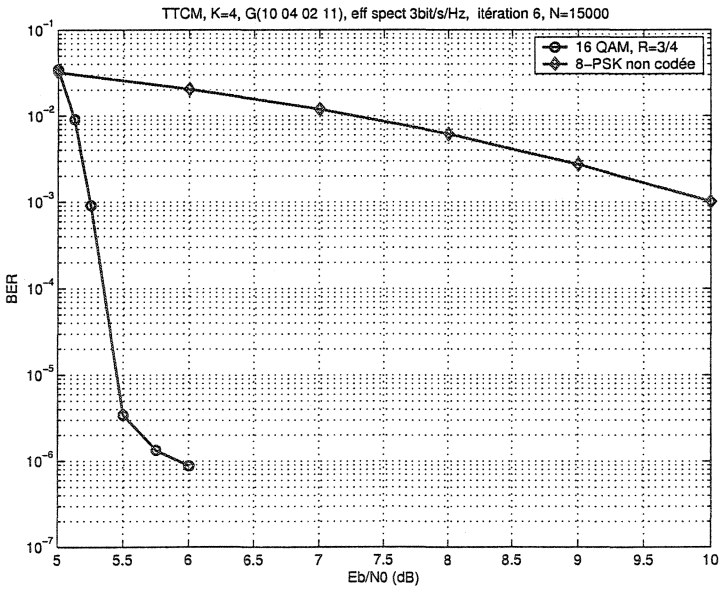

Figure 5.31: Gain de codage du schéma TTCM par rapport à la modulation non codée, efficacité spectrale 3 bits/s/Hz, $K = 4$, $N = 15000$

Nous avons recensé tous ces résultats dans le tableau 5.3, ce dernier recense les gains de codage obtenus par rapport au système non codé qui nous aurait permis d'avoir la même efficacité spectrale.

Tableau 5.3: Gain de codage par rapport à la modulation non codée à 10^{-5} pour $K = 4$

Taux de codage R	N	Ordre de la modulation	Gain à 10^{-5} / à la modulation non codée (dB)
2/3	900	8-PSK	3.5
3/4	900	16-QAM	6
2/3	2052	8-PSK	4.2
3/4	2052	16-PSK	3.5
3/4	2052	16-QAM	7
2/3	4158	8-PSK	4.6
3/4	4158	16-QAM	7.3

5.8 Influence de la longueur de l'entrelaceur

Nous avons remarqué que la longueur de l'entrelaceur influençait énormement les performances du codeur turbo pragmatique. Dans cette section nous allons étudier le comportement du codeur TTCM en fonction de la longueur de l'entrelaceur. Nous avons effectué des simulations avec 4 longueurs d'entrelaceurs différentes et pour 2 efficacités spectrales 2 et 3 bits/s/Hz. Ces résultats sont illustrés dans les figures 5.32 et 5.33. Le gain de codage est par contre moins important que le turbo pragmatique, on obtient un gain de codage de 1dB en passant d'un N=4096 à un N=256 pour le codeur TTCM par rapport à 2 dB pour le codeur turbo pragmatique. Pour des longueurs d'entrelaceur allant de 256 à 10000, on obtient un gain de codage d'environ 2dB. Malheureusement tout comme le turbo pragmatique l'augmentation

de la longueur de l'entrelaceur ne va pas sans une augmentation du délai du codage et du décodage.

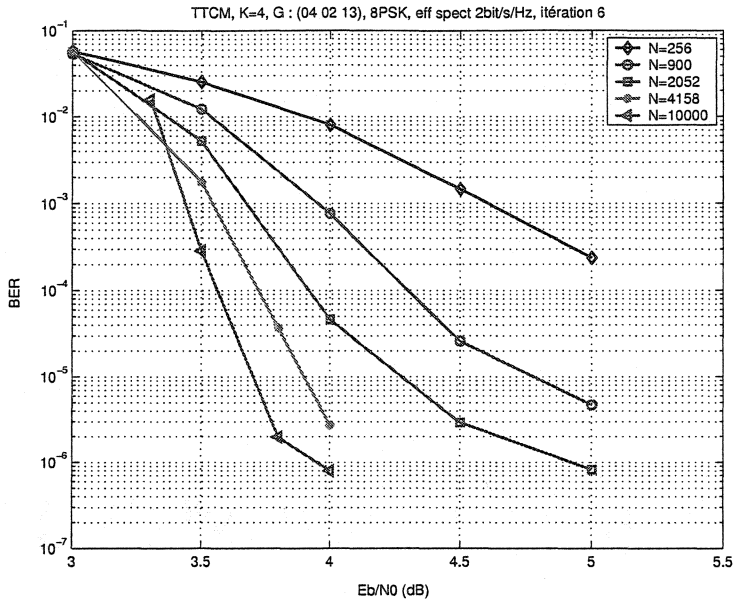

Figure 5.32: Influence de la longueur de l'entrelaceur sur les performances du schéma TTCM, $K = 4$, $G = (040213)$, 8-PSK

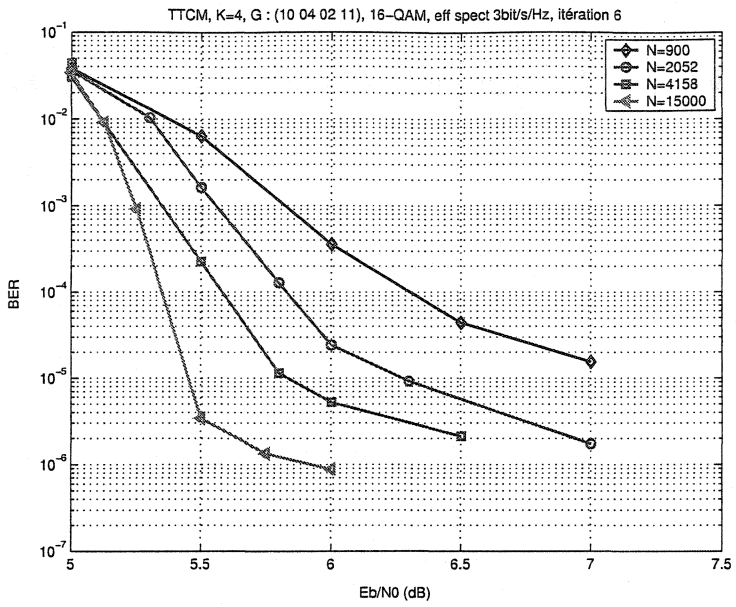

Figure 5.33: Influence de la longueur de l'entrelaceur sur les performances du schéma TTCM, $K = 4$, $G = (10040211)$, 16-QAM

5.9 Etude comparative du turbo pragmatique et du schéma TTCM

Nous avons étudié principalement deux schémas de modulations élévées : les schémas turbo pragmatique et TTCM. Le premier est pragmatique et donne de bonnes performances d'erreurs mais présente l'inconvénient d'avoir un décodage très lourd à effectuer. Le second n'est pas pragmatique mais est intéressant du fait de la modulation codée et donne également de bonnes performances d'erreurs. De plus il effectue un décodage basé directement sur les symboles reçus ce qui a pour avantage de reduire considérablement l'effort de calcul. Pour terminer ce chapitre, il serait utile de faire une étude comparative des deux derniers schémas de modulation. Nous avons effectué une étude comparative des performances d'erreur des deux schémas et de la complexité de mise en oeuvre. Afin de comparer les performances d'erreur des deux schémas, nous avons tracé des courbes montrées dans les figures 5.34 à 5.37.

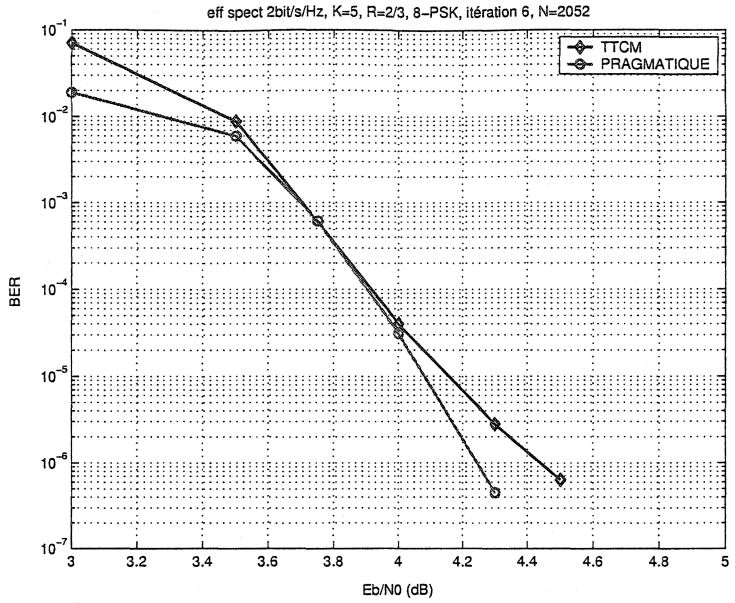

Figure 5.34: Comparaison des schémas TTCM et turbo pragmatique, efficacité spectrale 2 bit/s/Hz, $K = 5$, $N = 2052$, 8-PSK

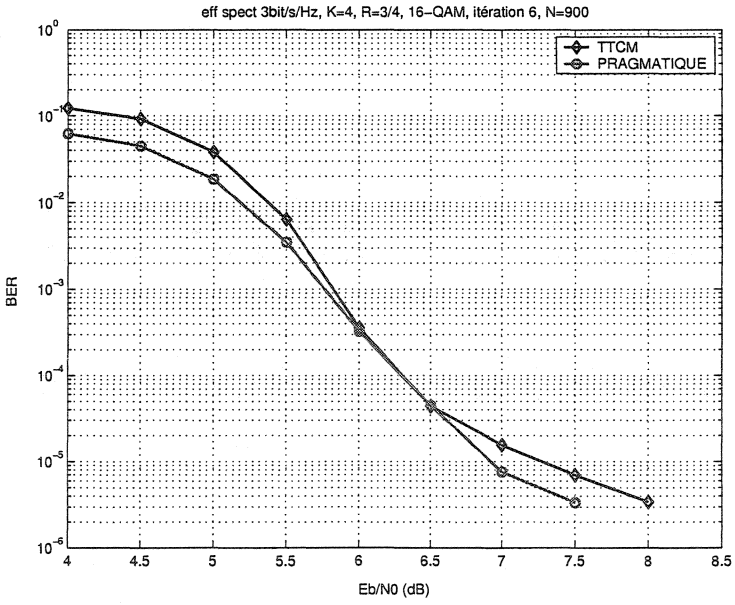

Figure 5.35: Comparaison des schémas TTCM et turbo pragmatique, efficacité spectrale 3 bit/s/Hz, $K = 4$, $N = 900$, 16-QAM

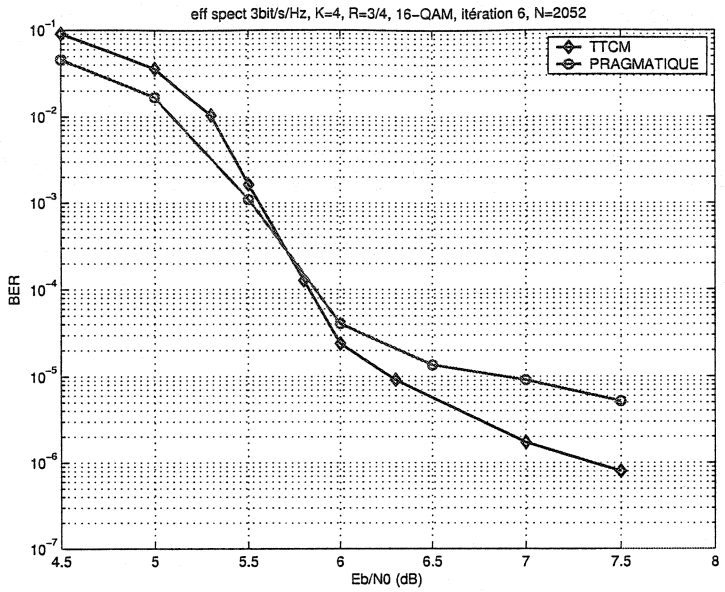

Figure 5.36: Comparaison des schémas TTCM et turbo pragmatique, efficacité spectrale 3 bit/s/Hz, $K = 4$, $N = 2052$, 16-QAM

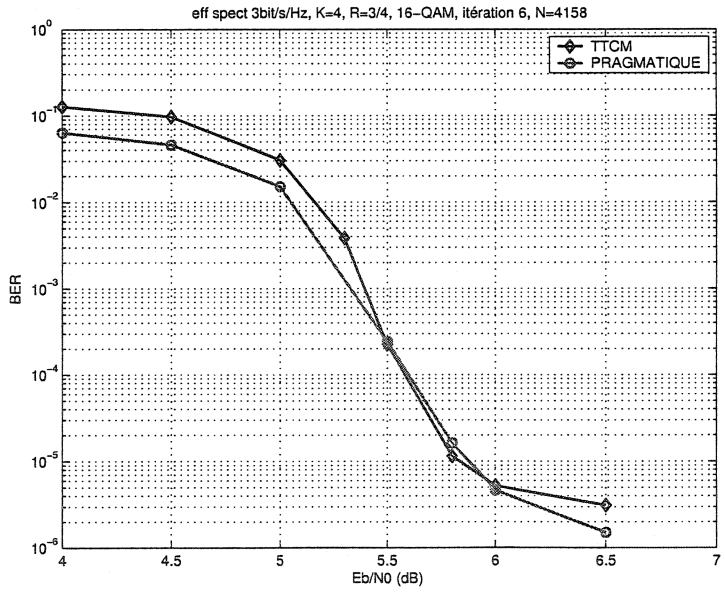

Figure 5.37: Comparaison des schémas TTCM et turbo pragmatique, efficacité spectrale 3 bit/s/Hz, $K = 4$, $N = 4158$, 16-QAM

Nous remarquons que pour les différentes longueurs de contraintes et d'entrelaceurs utilisées, les schémas TTCM et turbo pragmatique donnent pratiquement les mêmes performances d'erreur.

Le tableau 5.4 recense les calculs que nécessitent le codage et le décodage turbo pragmatique et TTCM, M est le nombre de signaux de la constellation utilisée et m, $log_2 M$. C représente tous les calculs par bit que demande un décodeur itératif turbo binaire.

Tableau 5.4: Comparaison de la complexité de l'algorithme de décodage TTCM et turbo pragmatique

	Nombre de calculs par bit			
	Codeur pragmatique	Codeur TTCM	Décodeur pragmatique	Décodeur TTCM
Exponentiel	-	-	$4 * N * M$	-
Multiplication	-	-	$4 * N * M$	-
Addition	-	-	$(4 * N * M) + 2N$	-
Division	-	-	N	-
Délai entrelaceur ou du désentrelaceur	-	$2 * N/m$	-	$(3N/m) + N$
Calculateur du LLR	-	-	$2 * C * N$	$2 * C * N/m$

Comme le montre le tableau 5.4, le décodeur turbo pragmatique a l'inconvénient d'intégrer beaucoup d'exponentielles à cause de la présence du module de calcul des LLR qui permet de reconstituer les bits à partir des symboles qui les ont constitués. Ce problème ne se pose pas avec le décodeur TTCM car les symboles sont directe-

ment traités. Le schéma turbo pragmatique est donc assez complexe à mettre en oeuvre mais a l'avantage d'être assez souple. On peut changer l'efficacité spectrale, la constellation utilisée et le taux de codage sans toutefois changer la structure du schéma ce qui n'est pas le cas pour le schéma TTCM, où pour passer à une autre constellation ou une autre efficacité spectrale il faut mettre en oeuvre une autre structure. Le schéma turbo pragmatique ou TTCM sera choisi en fonction du facteur qu'on désire privilégier, le pragmatisme ou la complexité de décodage. Pour pallier à la non pragmaticité du TTCM, nous avons proposé une structure dérivante: le schéma TTCM pragmatique.

5.9.1 Schéma TTCM pragmatique

Le schéma TTCM présente un inconvénient majeur: il n'est pas pragmatique. Implanter un système ayant plusieurs efficacités spectrales nécessite plusieurs codeurs. Pour palier à ce problème, nous allons nous servir des travaux de Viterbi, Wolf, Zehavi et Padovani [49]. Ces derniers ont introduit une méthode pragmatique pour aborder la modulation codée. Ils ont proposé un codeur convolutionnel de générateurs: $G_1 = 133$ et $G_2 = 171$, ayant 64 états et un taux de codage $R = 1/2$ associé à plusieurs constellations et où on ne code qu'un seul bit d'information. Pour avoir une efficacité spectrale de 1 bits/s/Hz, on n'utilise que 2 bits de parité. Pour une constellation 8-PSK, on se servira des deux bits de parité et d'un autre bit d'information qui lui ne sera pas codé, fournissant ainsi une efficacité spectrale de 2 bits/s/Hz. Lorsqu'il s'agit d'une constellation 16-PSK ou 16-QAM, on se servira des deux bits de parité et de deux autres bits d'informations qui n'auront pas été codés. Cette configuration nous donnera une efficacité spectrale de 3 bits/s/Hz. Le codeur TCM pragmatique est présenté dans la figure 5.38. Maintenant un codeur unique peut réaliser plusieurs types de modulation codée ayant différentes efficacités spectrales.

Le schéma TTCM pragmatique représenté à la figure 5.39 aura tout simplement la même structure que celle du TTCM à la seule différence qu'on aura des codeurs TCM pragmatiques à la place des codeurs TCM standards. Cependant cette approche pragmatique a une mémoire de 64 états, ce qui est nettement plus élevé par

rapport au schéma TTCM que nous avons jusque là étudié. En effet nous avons vu que la longueur de contrainte $K = 4$, c'est à dire avec une mémoire de 8 états, nous donnait d'aussi bons résultats que lorsque qu'on avait des longueurs de contrainte plus élévées.

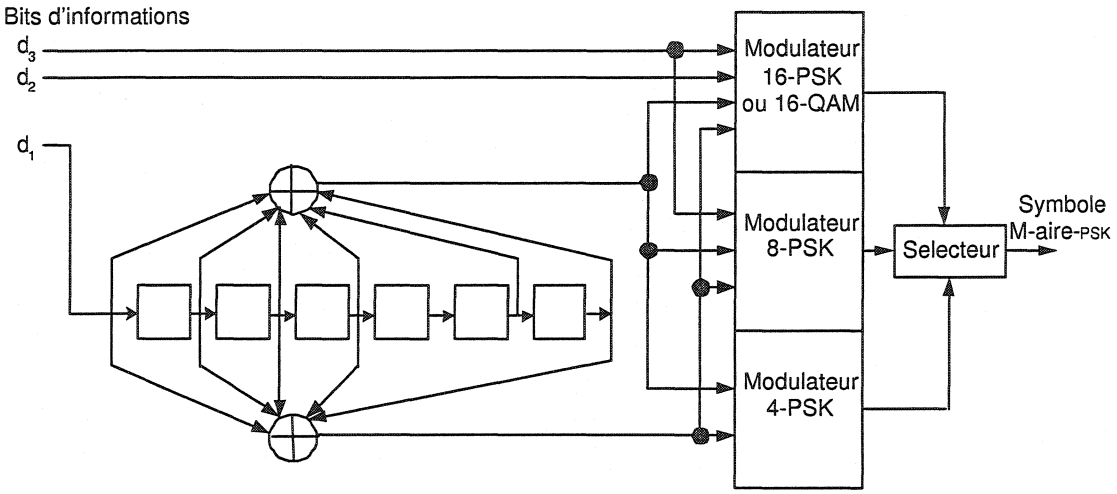

Figure 5.38: Codeur TCM pragmatique de Viterbi [11]

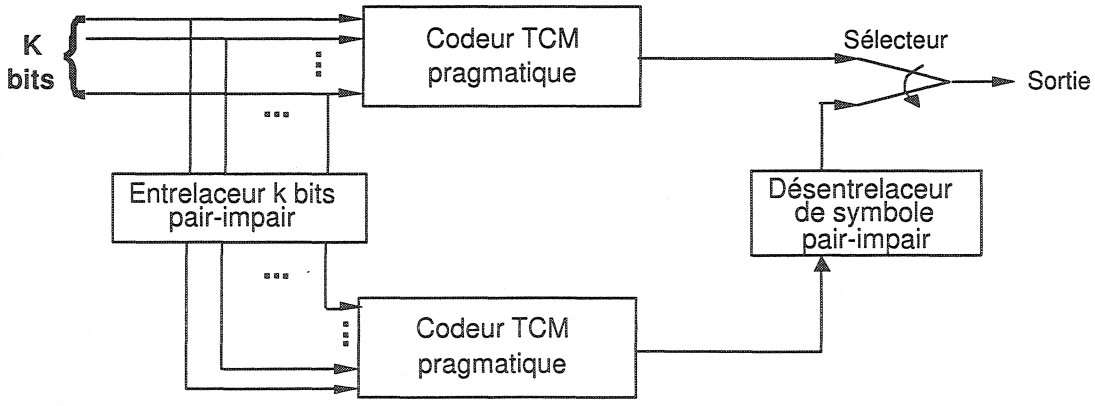

Figure 5.39: Codeur TTCM pragmatique

5.10 Conclusion

Dans ce chapitre nous avons étudié le schéma TTCM. Du point de vue performance d'erreur nous avons pratiquement les mêmes résultats que ceux obtenus avec le schéma turbo pragmatique. Le schéma TTCM a l'avantage d'effectuer une modulation codée et l'effort de calcul est nettement diminué par rapport au décodeur turbo pragmatique grâce à l'utilisation d'un décodeur par symbole. Il n'est par contre pas pragmatique car il demande un changement de structure pour avoir une efficacité spectrale différente. Pour pallier à ce problème nous avons proposé un schéma ayant la structure TTCM et qui nous permet de changer d'efficacité spectrale sans changer de schéma. L'étude de l'influence de la longueur de contrainte des codes nous a permis de remarquer les mêmes résultats que ceux qui ont été obtenus pour le schéma turbo pragmatique. Un code de longueur de contrainte $K = 3$ ne présente pas de bonnes performances d'erreur. Des codes de longueur de contrainte $K = 4$, 5 et 6 donnent pratiquement les mêmes résultats contrairement aux codes convolutionnels. Les performances d'erreur des codes turbo sont déterminées surtout par la distance effective libre au lieu de la distance libre pour les codes convolutionnels, et cette distance effective libre est pratiquement pareille pour les codes de longueur de contrainte K égale à 4, 5 et 6.

CHAPITRE 6

CONCLUSION ET PERSPECTIVES DE RECHERCHES

Dans ce mémoire nous nous sommes intéressés aux schémas de modulation élévée associés à la technique de codage turbo. Les codes turbo décrits pour la première fois en 1993 par Berrou, Glavieux et Thitimajshima présentent des performances d'erreur remarquables, leurs probabilités d'erreurs se rapprochant à 0.7dB de la limite théorique de Shannon. Malheureusement la modulation BPSK utilisée n'est pas très intéressante pour les applications à largeur de bande réduite. Nous avons donc étudié les schémas utilisant la technique turbo associée aux modulations multi-niveaux et ce sous un canal gaussien. Nous nous sommes intéréssés principalement à deux schémas de modulation que sont les schémas turbo pragmatique et TTCM. Nous avons commencé dans le chapitre 1 par présenter les bases d'un système de communications numériques, nous nous sommes ensuite attelé à présenter les codes convolutionnels qui sont un élément de base des codes turbo. Dans le chapitre 2 nous avons présenté les codes turbo et les différents éléments qui le caractérisent et en particulier les entrelaceurs et son décodage. Nous avons présenté le principe de décodage itératif turbo et détaillé les différents algorithmes qui existent dans la littérature et particulièrement l'algorithme MAP et son équivalent logarithmique, le LogMAP que nous avons utilisé pour nos simulations.

Une fois le principe de codage et décodage turbo acquis nous avons décrit les principaux schémas de modulations multi-niveaux qui existent dans la littérature pour ensuite faire une étude détaillée des schémas turbo pragmatique et TTCM. Ils ont fait respectivement l'objet des chapitre 4 et 5.

Nous avons commencé par étudier le schéma turbo pragmatique car c'est une structure très modulaire. On passe d'une efficacité spectrale, d'un taux de codage et d'une constellation à une autre sans pour autant modifier le schéma, malheureusement il demande un effort de calcul considérable. Une grande partie de l'étude a été faite sur l'influence de l'assignation des bits aux signaux de la constellation sur les performances d'erreur. Cette étude était nécessaire car il s'est avéré que d'une assignation à une autre on obtenait des différences de gains de codage allant jusqu'à 1dB. L'étude de ce schéma de modulation nous a permis de dégager une caractéristique remarquable des codes turbo et de tous les schémas utilisant le principe turbo : l'augmentation de la longueur de contrainte n'améliore pratiquement pas les performances d'erreur. On atteint au maximum un gain de codage de 0.3 dB d'une longueur de contrainte à la suivante. La longueur de contrainte présentant le meilleur compromis complexité de décodage performance d'erreur est égale à 4 aussi bien pour le schéma TTCM que pour le schéma turbo pragmatique. Un code de longueur de contrainte 3 n'est pas assez puissant pour présenter un interêt. Les longueurs de contrainte 5 et 6 quant à elles donnent les mêmes performances que $K = 4$. Nous avons par ailleurs effectué des simulations pour des efficacités spectrales allant de 2 jusqu'à 7 bits/s/Hz et étudié des constellations de signaux allant du 8-PSK au 256-QAM.

Le dernier chapitre a fait l'objet du schéma TTCM qui est relativement plus facile à mettre en oeuvre. Il intégre la modulation et le codage. Le principe de décodage multi-niveaux a été introduit. Ce principe nous a permis de diminuer énormement la latence en supprimant le module de calcul du LLR. Il donne de bonnes performances d'erreurs pratiquement semblables à celles obtenues pour le schéma turbo pragmatique. Cependant il n'est pas pragmatique et changer d'efficacité spectrale ou de constellation revient à concevoir un autre système. Pour pallier à ce problème nous avons proposé un schéma TTCM pragmatique. Pour finir nous avons effectué une étude comparative de ces deux schémas de modulation.

Nous avons présenté trois principaux schémas de modulation multi-niveaux à savoir le schéma multi-niveaux, le schéma turbo pragmatique et le schéma

TTCM. Le premier a tout juste été présenté et n'a pas fait l'objet d'une étude plus approfondie car il est trop complexe à mettre en oeuvre sans pour autant que ses performances d'erreur soient meilleures à celles des deux autres schémas de modulation multi-niveaux. Les schémas turbo pragmatique et TTCM donnent pratiquement les mêmes performances d'erreur pour des codes de longueurs de contrainte pareilles et pour une même constellation de signaux. Le schéma turbo pragmatique est assez modulaire mais son algorithme de décodage est assez complexe. L'algorithme de décodage du schéma TTCM est relativement plus simple mais ce schéma n'est pas modulaire. Ce problème a été résolu par la présentation du schéma TTCM pragmatique, il serait donc plus intéressant de l'utiliser.

Ce mémoire nous a permis d'effectuer une étude approfondie des schémas de modulations élévées associés à la technique de codage turbo sous un canal gaussien. Il fournit également de nouvelles bases pour des recherches futures. Voici une liste exhaustive de recherches intéressantes :

- Etude des schémas de modulations élévées associés à la technique de codage turbo sous un canal de Rayleigh.
- Association du codage TTCM et du codage spatio-temporel.
- Recherche des meilleurs codes TCM pour les modulations 32-QAM, 64-QAM, 128-QAM et 256-QAM.
- Etude d'autres schémas de modulation qui réduiraient l'effort de calcul du turbo pragmatique tout en conservant son caractère pragmatique.

RÉFÉRENCES

[1] Bahl L., Cocke J., Jelinek F. et Raviv J., "Optimal Decoding of Linear Codes For Minimizing Symbol Error Rate", *IEEE Trans. Inform. Theory*, Vol. 20, pp. 284-287, Mars 1974.

[2] Battail G., Berrou C. et Glavieux A., "Pseudo-Random Recursive Convolutional Coding For Near-Capacity Performance", *IEEE Global Telecommunications Conference*, Vol.4, Houston, pp. 23-27, 1993.

[3] Benedetto S., Divsalar D., Pollara F. et Montorsi G., "Bandwith Efficient Parallel Concatenated Coding Schemes", *Electronic Letters*, Vol.31, no.24, pp. 2067-2069, Novembre 1995.

[4] Benedetto S., Divsalar D., Pollara F. et Montorsi G., "Parallel Concatenated Trellis Coded Modulation", *Proc. ICC'96*, pp. 974-978, Juin 1996.

[5] Benedetto S. et Montorsi G., "Performance Evaluation Of Parallel Concatenated Codes", *Proceedings of ICC'95*, Vol.2, Seattle, pp. 663-667, Juin 1995.

[6] Benedetto S. et Montorsi G., "Unveiling Turbo Codes: Some Results On Parallel Concatenated Coding Schemes", *IEEE Trans. Inform. Theory*, Vol.42, no.2, pp. 409-428, Mars 1996.

[7] Berrou C., Glavieux A. et Thitimasjshima P., "Near Shannon Limit Error Correcting Coding And Decoding: Turbo Codes", *Proceedings of ICC'93*, pp. 1064-1070, 1993.

[8] Berrou C., LeGoff S. et Glavieux A., "Turbo Codes and High Efficiency Modulation", *Proceedings of ICC'94*, New Orleans, pp. 645-649, Mai 1994.

[9] Blackert. W.J., "Implementation Issues Of Turbo Trellis Coded Modulation", Thèse de Doctorat, *Université de Virginie*, Charlottesville, Mai 1996.

[10] Blackert. W.J. et Wilson S.G., "Turbo Trellis Coded Modulation", *CISS'96*, Université de Virginie, Charlottesville, Mai 1996.

[11] Chan F., "Décodage Simplifié : Algorithme Adaptatif Pour Codes Convolutionnels Et Techniques De Perforation Pour Modulation Codée", Thèse de Doctorat, *Ecole Polytechnique*, Montréal, 1994.

[12] Chi D. T., "A New Block Helical Interleaver", *MILCOM 1992*, San Diego, CA, pp. 799-804, Octobre 1992.

[13] Chi D. T., "Helical Interleavers", *Proceedings Of The International Telemetry Conference*, Vol. XXIV, pp. 465-472, Octobre 1990.

[14] Divsalar D. et Dolinar S., "Weight Distributions For Turbo Codes Using Random And Nonrandom Permutations", *JPL TDA Progress Report*, 42-122, Pasadena, CA, pp.56-65, Août 1995.

[15] Divsalar D. et McEliece R.J., "Effective Free Distance Of Turbo Codes", *Electronics Letters*, Vol. 32, no.5, pp. 445-446, Fevrier 1996.

[16] Divsalar D. et Pollara F., "Turbo Codes For Deep-Space Communications", *TDA Progress Report 42-120*, pp. 29-39, Fevrier. 1995.

[17] Divsalar D. et Pollara F., "Turbo Codes For PCS Applications", *Proceedings of ICC'95*, Seattle, WA, pp. 59-59, Juin 1995.

[18] Elias P., "Error-Free Coding", *IEEE Trans. Inform. Theory*, Vol. 4, pp. 29-37, 1954.

[19] Erfanian J. A., Pasupathy S. et Gulot G., "Reduced Complexity Symbol Detectors With Parallel Structures For ISI Channels", *IEEE Trans. Commun.*, Vol. 42, pp. 1661-1671, 1994.

[20] Fano R. M., "A Heuristic Discussion of Probabilistic decoding", *IEEE Trans on Inform. Theory*, Vol. 9, pp. 64-74, Avril 1963.

[21] Forney G., "Convolutional Codes: Algebraic Structure", *IEEE Transactions On Information Theory*, Vol. 16, pp. 720-738, Novembre. 1970.

[22] Haccoun D., "Notes de Cours ELE 6703, Théorie Des Communications", *Ecole Polytechnique Montréal*, Hiver 2003.

[23] Haccoun D., Bhargava V.K., Matyas R. et Nuspl P.P, "Digital Communications By Satellite", *John Wiley and Sons*, New York, 1981.

[24] Hagenauer J. et Hoeher P., "A Viterbi Algorithm With Soft-Decision Outputs And Its Applications", *IEEE Globecom*, pp. 1680-1686, Novembre. 1989.

[25] Hagenauer J., Offer E. et Papke L., "Iterative Decoding Of Binary Block And Convolutional Codes", *IEEE Trans. on Inform. Theory*, Vol. 42, pp. 429-445, Mars 1996.

[26] Haykin S., "Digital Communications", *John Wiley & Sons*, 1988.

[27] Heegard C. et Wicker S.B., "Turbo Coding", *Kluwer Academic Publisher*, 1999.

[28] Herzberg H., " Multilevel Turbo Coding With Short Interleavers", *IEEE Journal On Selected Areas In Communications*, pp. 303.309, Fevrier 1998.

[29] Imai H., Hirakawa S., "A New Multilevel Coding Method Using Error Correcting Codes", *IEEE Transactions on Information Theory*, pp. 371-377, Mai 1977.

[30] Jelinek F., "A Fast Sequential Decoding Algorithm Using A Stack", *IBM Journal Research and Development*, Vol. 13, pp. 675-685, Novembre. 1969.

[31] Khaled L., "Etude Des Performances Des Codes Turbo", Mémoire de Maîtrise, *Ecole Polytechnique*, Montréal, Juin 2001.

[32] LeGoff S., "Les Turbo Codes Et Leur Application Aux Transmissions A Forte Efficacité Spectrale", Thèse de Doctorat, *Université de Bretagne Occidentale*, Paris, Mai 1994.

[33] Massey J. L. et Sain M. K., "Inverses Of Linear Sequential Cicuits", *IEEE Trans. Comput.*, C-17, pp. 330-337, Avril 1968.

[34] Parron J., "Décodage Des Codes Turbo Par L'algorithme MAP:Simplifications Et Applications Aux Modulations Multi-Niveaux", Mémoire de Maîtrise, *Ecole Polytechnique*, Montréal, Juillet 2002.

[35] Perez C.L., Seghers J. et Costello D.J.,"A Distance Spectrum Interpretation Of Turbo Codes", *IEEE Trans. on Inform. Theory*, Vol.42, no.6, pp. 1698-1709, Novembre 1996.

[36] Pietrobon S.S, "Implementation And Performance Of A Turbo/MAP Decoder", *Int.J.Satellite Commun*, Fevrier.1997.

[37] Ramsey J. L., "Realization Of Optimum Interleavers", *IEEE Transactions on Information Theory*, IT-16, pp. 338-346, Mai 1970.

[38] Reed M., Asenstofer J., "A Novel Variance Estimator for Turbo-Code Decoding", *International Conference on Telecommunications*, pp. 173-178, Avril 1997.

[39] Reed M. et Pietrobon S., "Turbo-Code Termination Schemes And A Novel Alternative For Short Frames", *PIMRC 96*, Vol. 2, pp. 354-358, Octobre. 1996.

[40] Robertson P., Villebrun E. et Hoeher P., "A Comparison Of Optimal And Sub-Optimal MAP Decoding Algorithms Operating In The Log Domain", *Proceedings of ICC'95*, pp. 1009-1013, 1995.

[41] Robertson P., Worz T., "Bandwith-Efficient Turbo Trellis-Coded Modulation Using Punctured Component Codes", *IEEE Journal On Selected Areas In Communications*, pp. 206-218, Fevrier 1998.

[42] Royer G., "Evaluation Des Entrelaceurs Au Sein Des Codes Turbo Par Simulations", Mémoire de Maîtrise, *Ecole Polytechnique*, Montréal, 2000.

[43] Shao R., Lin S. et Fossorier M., "Two Simple Stopping Criteria For Turbo Decoding", *IEEE Trans. On Commun.*, Vol. 47, pp. 1117-1120, Août 1999.

[44] Thitimajshima P., "Les Codes Convolutifs Récursifs Systématiques Et Leur Application A La Concaténation Parallèle", Thèse de Doctorat, *Université de Bretagne Occidentale*, Paris, Décembre 1993.

[45] Ungerboeck G., "Channel Coding With Multilevel/Phase Signals", *IEEE Trans. On Inform Theory*, Vol.IT-28, pp. 55-66, Janvier 1982.

[46] Ungerboeck G., "Trellis Coded Modulation With Redundant Signal Sets Part II : State Of The Art", *IEEE Communications Magazine*, Vol.25, no.2, pp. 12-21, Fevrier 1987.

[47] Ungerboeck G. et Csajka I., "On The Improving Data-link Performance By Increasing The Channel Alphabet And Introducing Sequence Coding", *Int. Symp. Inform. Theory*,Ronneby , Juin 1976.

[48] Viterbi A.J., "Error Bounds For Convolutional Codes And An Asymptotically Optimun Decoding Algorithm", *IEEE Trans. On Inform Theory*, Vol.IT-13, pp. 260-269, Avril 1979.

[49] Viterbi A.J., Wolf J.K., Zehavi E. et Padovani R., "A Pragmatic Approach To Trellis Coded Modulation", *IEEE Communications Magazine*, pp. 11-19, Juillet 1989.

[50] Vucetic, B. et Yuan, J., "Turbo codes:Principles And Applications", *Kluwer Academic Publisher, Boston*, 2000.

[51] Wachsmann U. et Huber J., "Power And Bandwidth Efficient Digital Communication Using Turbo Codes In Multilevel Codes", *European Transaction on Telecommuncations*, pp. 557-567, 1995.

[52] Wachsmann U., Huber J. et Fischer R.F.H., "Multilevel Codes: Theoretical Concepts And Practical Design Rules", *IEEE Transactions on Information Theory*, pp. 1361-1391, Juillet 1999.

[53] Wilson S.G., "Digital Modulation And Coding", *Englewood Cliffs, New Jersey : Prentice-Hall*, 1995.

[54] Wozencraft M. et Reiffen B., "Sequential Decoding", *MIT Press*, Cambridge, Mass., 1951.

[55] Zerong Y., "Analyse Et Performance Des Codes Convolutionnels Recursifs Systématiques", Mémoire de Maîtrise, *Ecole Polytechnique*, Montréal, 1998.

Annexe I

Développement des métriques de l'algorithme MAP

I.1 Métrique d'état en avant

Il s'agit de développer une relation entre la métrique d'état $\alpha_k^i(m)$ et $\alpha_{k-1}^i(m)$. De (3.14) nous avons la probabilité $\alpha_k^i(m)$ qui est égale à : $P(d_k = i, S_k = m, \mathbf{R}_1^k)$. Développons cette équation :

$$\begin{aligned}
\alpha_k^i(m) &= P(d_k = i, S_k = m, \mathbf{R}_1^k) \\
&= \sum_{m'=0}^{M-1} \sum_{j=0}^{1} P(d_k = i, d_{k-1} = j, S_k = m, S_{k-1} = m', \mathbf{R}_k, \mathbf{R}_1^{k-1}) \\
&= \sum_{m'=0}^{M-1} \sum_{j=0}^{1} P(d_{k-1} = j, S_{k-1} = m', \mathbf{R}_1^{k-1}). \\
&\quad P(d_k = i, S_k = m, \mathbf{R}_k | d_{k-1} = j, S_{k-1} = m', \mathbf{R}_1^{k-1}) \qquad \text{(I.1)}
\end{aligned}$$

En tenant du fait que les évènements après l'instant (k-1) ne sont pas influencés par la séquence reçue $\mathbf{R}_1^{k-1}$, si le bit d_{k-1} et l'état S_{k-1} sont connus alors l'expression de $\alpha_k^i(m)$ devient:

$$\begin{aligned}
\alpha_k^i(m) &= \sum_{m'=0}^{M-1} \sum_{j=0}^{1} P(d_{k-1} = j, S_{k-1} = m', \mathbf{R}_1^{k-1}). \\
&\quad P(d_k = i, S_k = m, \mathbf{R}_k | d_{k-1} = j, S_{k-1} = m') \qquad \text{(I.2)}
\end{aligned}$$

Rappelons que de (3.16) nous avons:

$$\gamma_i(\mathbf{R}_k, m', m) = P(d_k = i, S_k = m, \mathbf{R}_k | d_{k-1} = j, S_{k-1} = m') \qquad \text{(I.3)}$$

De (I.2) et (I.3), nous obtenons une expression récursive pour $\alpha_k^i(m)$:

$$\alpha_k^i(m) = \sum_{m'=0}^{M-1} \sum_{j=0}^{1} \gamma_i(\mathbf{R}_k, m', m) \alpha_{k-1}^j(m') \tag{I.4}$$

I.2 Métrique d'état en arrière

Dans le cas de la métrique d'état en arrière, il s'agit de développer une relation entre la métrique $\beta_k^i(m)$ et $\beta_{k+1}^i(m)$. (3.15) nous donne l'expression de la probabilité $\beta_k^i(m)$ qui est égale à: $P(\mathbf{R}_{k+1}^N | d_k = i, S_k = m)$. Développons cette équation:

$$\begin{aligned}
\beta_k^i(m) &= P(\mathbf{R}_{k+1}^N | d_k = i, S_k = m) \\
&= \sum_{m'=0}^{M-1} \sum_{j=0}^{1} P(d_{k+1} = j, S_{k+1} = m', \mathbf{R}_{k+1}^N | S_k = m, d_k = i) \\
&= \sum_{m'=0}^{M-1} \sum_{j=0}^{1} P(d_{k+1} = j, S_{k+1} = m', \mathbf{R}_{k+1}, \mathbf{R}_{k+2}^N | S_k = m, d_k = i) \\
&= \sum_{m'=0}^{M-1} \sum_{j=0}^{1} (\mathbf{R}_{k+2}^N | d_{k+1} = j, S_{k+1} = m', \mathbf{R}_{k+1}, S_k = m, d_k = i). \\
&\quad \frac{P(d_{k+1} = j, S_{k+1} = m', \mathbf{R}_{k+1}, S_k = m, d_k = i)}{Pr(S_k = m, d_k = i)} \\
&= \sum_{m'=0}^{M-1} \sum_{j=0}^{1} Pr(\mathbf{R}_{k+2}^N | S_{k+1} = m', d_{k+1} = j). \\
&\quad P(d_{k+1} = j, S_{k+1} = m', \mathbf{R}_{k+1} | S_k = m, d_k = i) \\
&= \sum_{m'=0}^{M-1} \sum_{j=0}^{1} \beta_{k+1}^j(m') \gamma_j(\mathbf{R}_{k+1}, m, m')
\end{aligned} \tag{I.5}$$

Il reste maintenant à simplifier davantage la métrique d'état $\gamma_i(\mathbf{R}_k, m', m)$.

I.3 Métrique d'état

De (3.16) nous avons :

$$\begin{aligned}\gamma_i(\mathbf{R}_k, m', m) &= P(d_k = i, S_k = m, \mathbf{R}_k | d_{k-1} = j, S_{k-1} = m') \\ &= P(\mathbf{R}_k | d_{k-1} = j, S_{k-1} = m', d_k = i, S_k = m). \\ &\quad P(d_k = i | d_{k-1} = j, S_{k-1} = m', S_k = m). \\ &\quad P(S_k = m | d_{k-1} = j, S_{k-1} = m') \end{aligned} \tag{I.6}$$

Le codeur turbo émet 2 bits : le bit systématique x_k et le bit de parité y_k, donc $\mathbf{R}_k = (x_k, y_k)$. Conditionnellement sur $(d_{k-1} = j, S_{k-1} = m', d_k = i, S_k = m)$, x_k et y_k sont 2 variables non corrélées, l'expression $\gamma_i(\mathbf{R}_k, m', m)$ devient donc:

$$\begin{aligned}\gamma_i(\mathbf{R}_k, m', m) &= P(x_k | d_{k-1} = j, S_{k-1} = m', d_k = i, S_k = m). \\ &\quad P(y_k | d_{k-1} = j, S_{k-1} = m', d_k = i, S_k = m). \\ &\quad P(d_k = i | d_{k-1} = j, S_{k-1} = m', S_k = m). \\ &\quad P(S_k = m | d_{k-1} = j, S_{k-1} = m') \end{aligned} \tag{I.7}$$

La connaissance $d_k = i$ et $S_k = m$ définit complètement le chemin à travers le treillis donc l'expression de $\gamma_i(\mathbf{R}_k, m', m)$ peut se simplifier commme suit:

$$\begin{aligned}\gamma_i(\mathbf{R}_k, m', m) &= P(x_k | d_k = i, S_k = m). \\ &\quad P(y_k | d_k = i, S_k = m). \\ &\quad P(d_k = i | d_{k-1} = j, S_{k-1} = m', S_k = m). \\ &\quad P(S_k = m | d_{k-1} = j, S_{k-1} = m') \end{aligned} \tag{I.8}$$

Compte tenu du fait que la modulation effectuée est une modulation BPSK et que le bruit du canal affecte les bits transmis nous avons $x_k = 2d_k - 1 + n_k$, et $y_k = 2c_k - 1 + n_k$, c_k étant le bit de parité. Si le bruit affectant le canal suit une distribution de Rayleigh, x_k et y_k seront respectivement deux variables aléatoires

de Rayleigh de moyennes $m_\alpha(2d_k - 1)$ et $m_\alpha(2c_k - 1)$. On a donc :

$$P(x_k|d_k = i, S_k = m) = \frac{1}{\sqrt{2\pi\sigma^2}}\, exp(-\frac{m_\alpha(x_k - (2i-1))^2}{2\sigma^2}) \tag{I.9}$$

$$P(y_k|d_k = i, S_k = m) = \frac{1}{\sqrt{2\pi\sigma^2}}\, exp(-\frac{m_\alpha(y_k - (2c_k-1))^2}{2\sigma^2}) \tag{I.10}$$

$P(S_k = m|d_{k-1} = j, S_{k-1} = m')$ vaut 0 ou 1 selon qu'il existe ou non une transition dans le treillis entre l'état $S_k = m$ et $S_{k-1} = m'$, pour un bit $d_{k-1} = j$ reçu.

$P(d_k = i|d_{k-1} = j, S_{k-1} = m', S_k = m)$ dépend de la probabilité à priori. L'information à priori $L_{a,k}$ est définie comme:

$$L_{a,k} = ln\left(\frac{P(d_k = 1)}{P(d_k = 0)}\right) \tag{I.11}$$

En utilisant l'égalité $P(d_k = 1) = 1 - P(d_k = 0)$ on peut dire :

$$P(d_k = i|L_{a,k}) = \frac{exp(i.L_{a,k})}{1 + exp(L_{a,k})} \tag{I.12}$$

$$P(d_k = 1|L_{a,k}) = \frac{exp(L_{a,k})}{1 + exp(L_{a,k})} \tag{I.13}$$

$$P(d_k = 0|L_{a,k}) = \frac{1}{1 + exp(L_{a,k})} \tag{I.14}$$

On a donc:

$$P(d_k = i|d_{k-1} = j, S_{k-1} = m', S_k = m) = \frac{exp(i.L_{a,k})}{1 + exp(L_{a,k})} \tag{I.15}$$

Pour la première itération de décodage, cette probabilité vaut 1/2, car la probabilité d'avoir 0 ou 1 est la même.

Des équations (I.8), (I.9), (I.10) et (I.12), l'expression de $\gamma_i(\mathbf{R}_k, m', m)$ devient:

$$\gamma_i(\mathbf{R}_k, m', m) = \frac{1}{2\pi\sigma^2} e^{-\frac{m_\alpha(x_k-(2i-1))^2}{2\sigma^2}} e^{-\frac{m_\alpha(y_k-(2c_k-1))^2}{2\sigma^2}} \cdot \frac{e^{iL_{a,k}}}{1+e^{L_{a,k}}} P(S_k = m | d_{k-1} = j, S_{k-1} = m') \tag{I.16}$$

On rappelle que $P(S_k = m | d_{k-1} = j, S_{k-1} = m')$ vaut 0 ou 1 selon qu'il existe ou non une transition dans le treillis entre l'état $S_k = m$ et $S_{k-1} = m'$, pour un bit $d_{k-1} = j$ reçu.

Pour un canal gaussien m_α est tout simplement égal à 1.

Annexe II

Résultats supplémentaires chapitre4

Nous présentons dans cet annexe les courbes relatives à l'influence de l'assignation des bits à la constellation, de l'efficacité spectrale et de la longueur de contrainte sur la probabilité d'erreur. Tous ces résultats ont été obtenus par simulation à l'ordinateur.

II.1 Influence de l'assignation des bits à la constellation

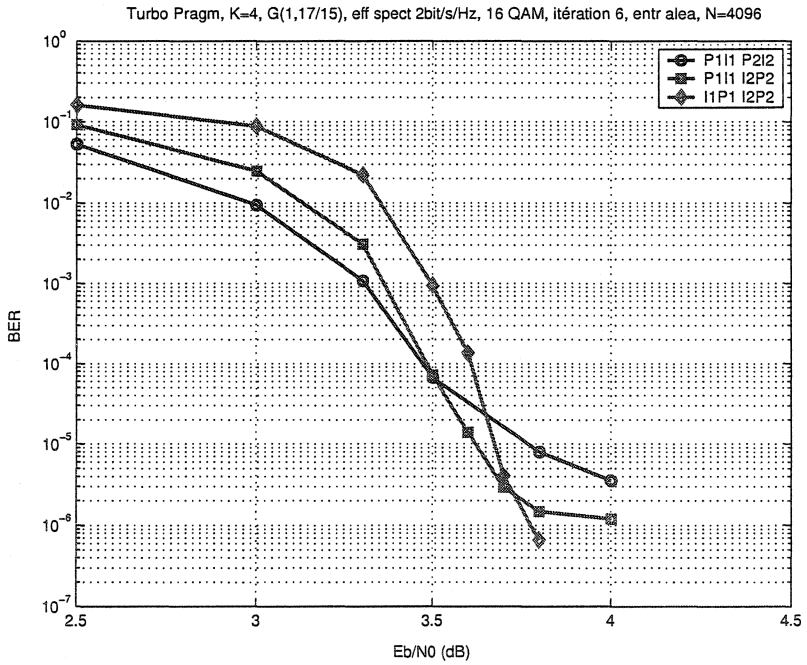

Figure II.1: Influence de l'assignation sur les performances du schéma turbo pragmatique, K=4, G=(1,17/15), N=4096, efficacité spectrale 2 bits/s/Hz, 16-QAM

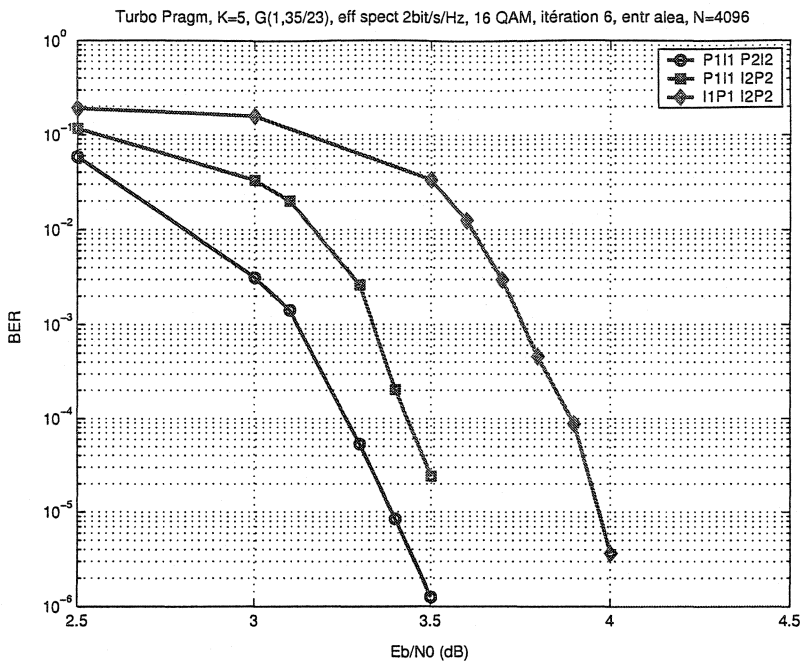

Figure II.2: Influence de l'assignation sur les performances du schéma turbo pragmatique, K=5, G=(1,35/23), N=4096, efficacité spectrale 2 bits/s/Hz, 16-QAM

II.2 Influence de l'efficacité spectrale

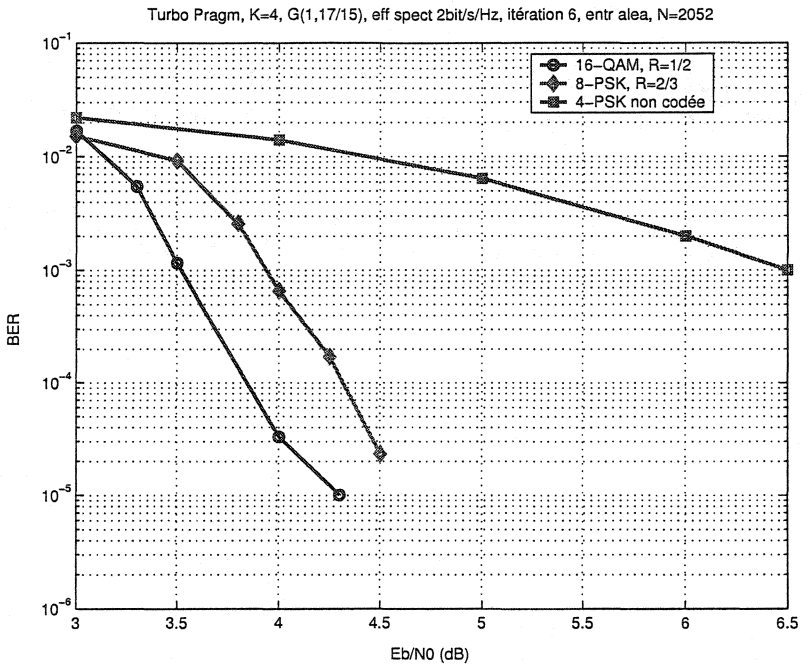

Figure II.3: Codeur turbo pragmatique, K=4, G=(1,17/15), N=2052, efficacité spectrale 2 bits/s/Hz

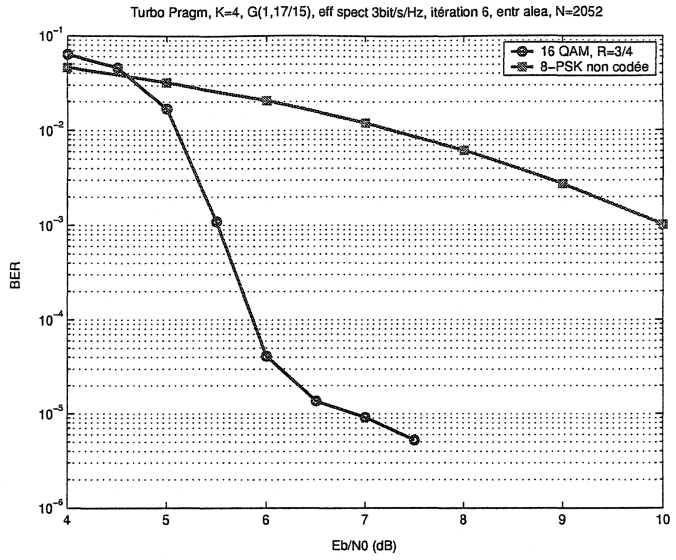

Figure II.4: Codeur turbo pragmatique, K=4, G=(1,17/15), N=2052, efficacité spectrale 3 bits/s/Hz

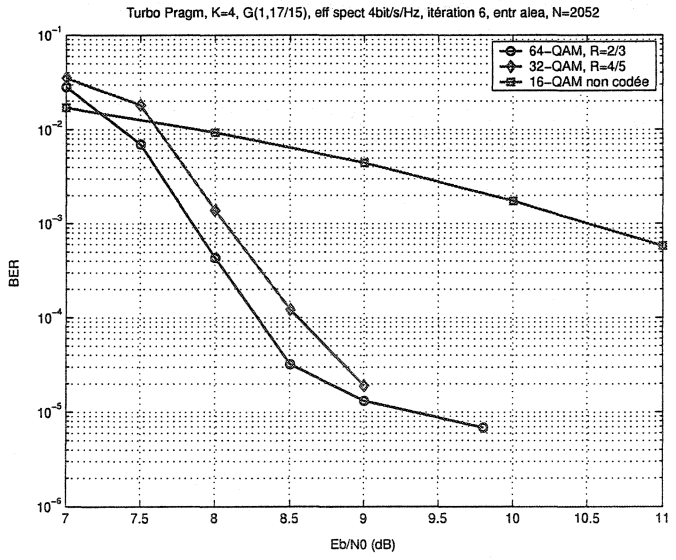

Figure II.5: Codeur turbo pragmatique, K=4, G=(1,17/15), N=2052, efficacité spectrale 4 bits/s/Hz

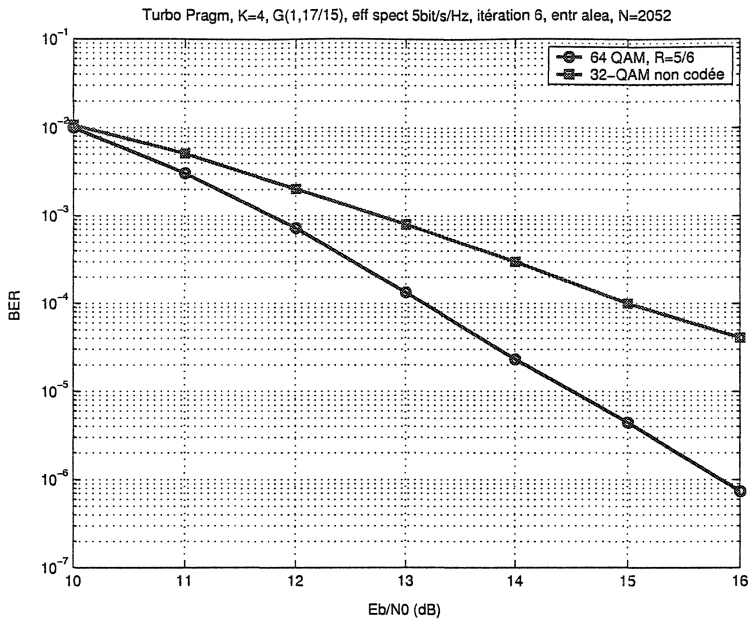

Figure II.6: Codeur turbo pragmatique, K=4 G=(1,17/15), N=2052, efficacité spectrale 5 bits/s/Hz

II.3 Influence de l'efficacité spectrale sur l'influence de l'assignation des bits à la constellation

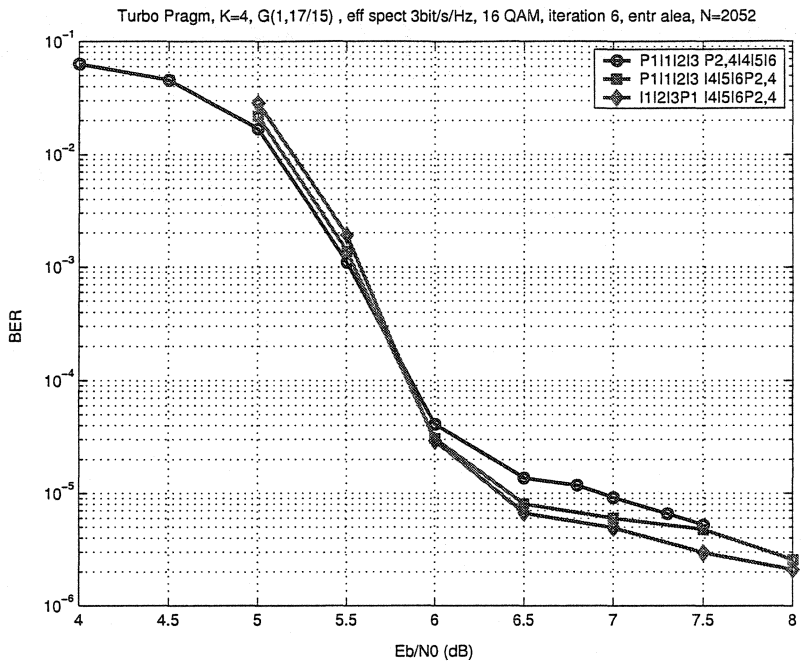

Figure II.7: Influence de l'efficacité spectrale sur l'assignation des bits, K=4, G=(1,17/15), N=2052, efficacité spectrale 3bits/s/Hz, codeur turbo pragmatique

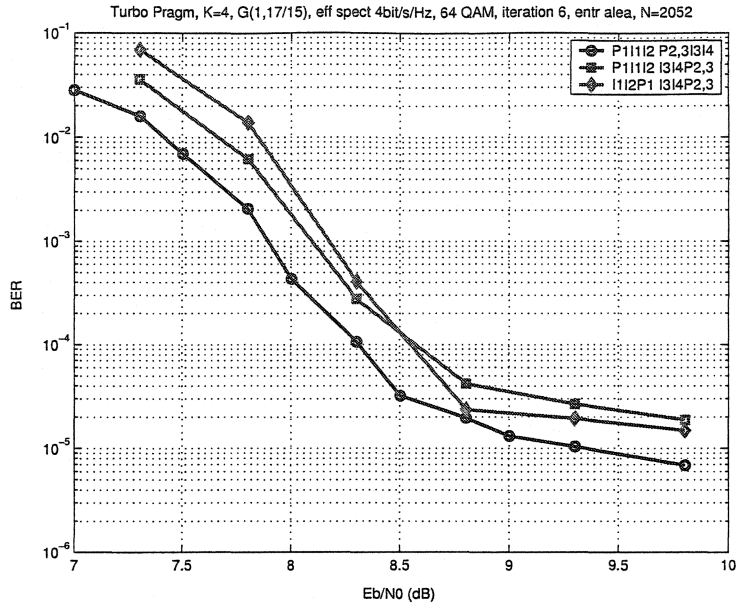

Figure II.8: Influence de l'efficacité spectrale sur l'assignation des bits, K=4, G=(1,17/15), N=2052, efficacité spectrale 4bits/s/Hz, codeur turbo pragmatique

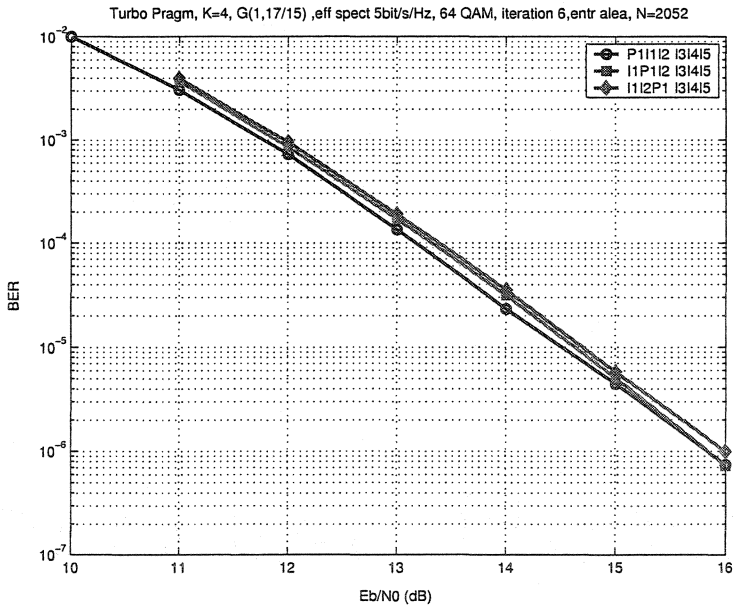

Figure II.9: Influence de l'efficacité spectrale sur l'assignation des bits, K=4, G=(1,17/15), N=2052, efficacité spectrale 5bits/s/Hz, codeur turbo pragmatique

II.4 Influence de la longueur de contrainte

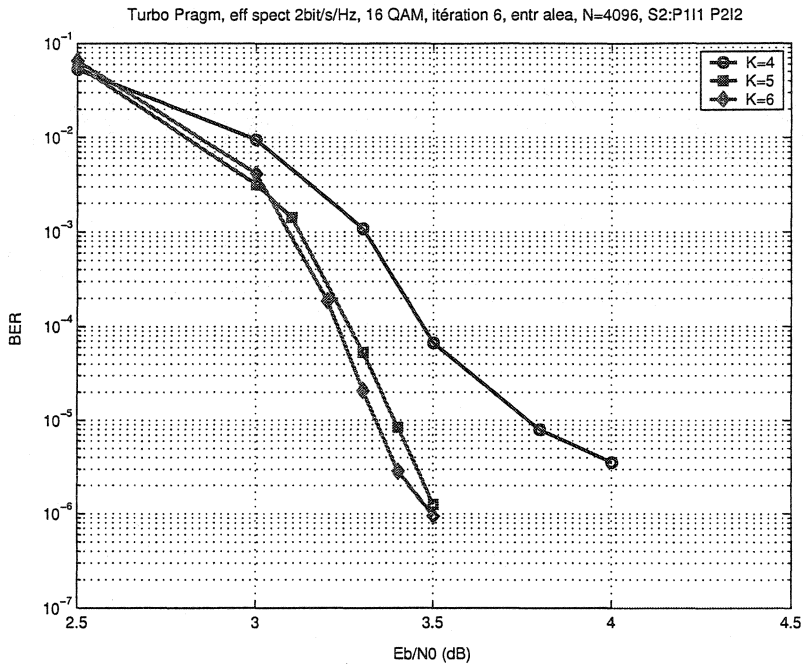

Figure II.10: Influence de la longueur de contrainte, codeur turbo pragmatique, N=4096, itération 6, 16 QAM

Annexe III

Résultats supplémentaires chapitre5

Nous présentons dans cet annexe les courbes relatives à l'étude du schéma TTCM. Nous avons étudié l'influence du partionnement sur les performances du schéma TTCM et recensé les gains de codage obtenus par le codage TTCM par rapport à la modulation non codée pour différentes efficacités spectrales.

III.1 Influence de l'assignation selon Ungerboeck et selon Gray des bits à la constellation sur les performances du schéma TTCM

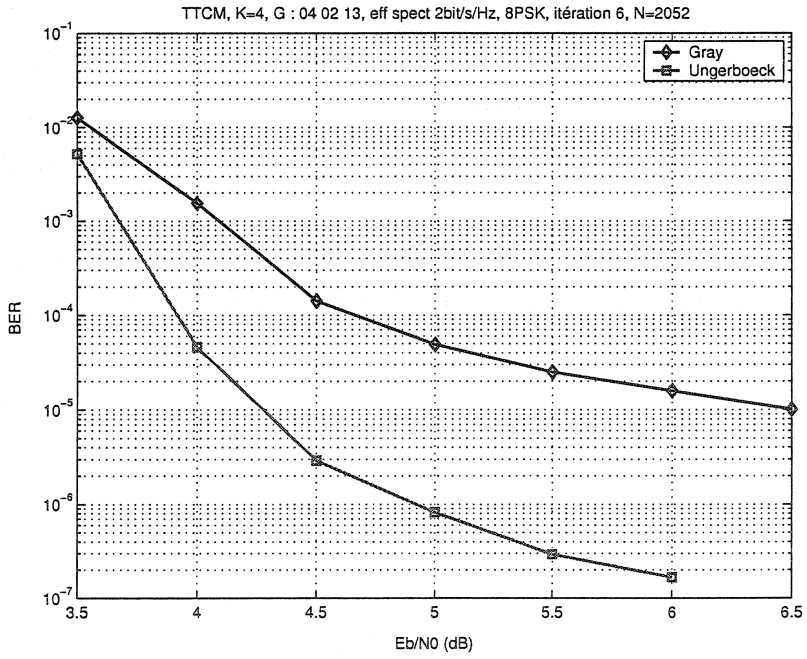

Figure III.1: Comparaison de l'assignation selon Ungerboeck et selon Gray des bits à la constellation sur les performances du schéma TTCM, 8-PSK, N=2052, K=4, G : 04 02 13

III.2 Gain de codage obtenu par le codage TTCM par rapport à la modulation non codée pour différentes efficacités spectrales

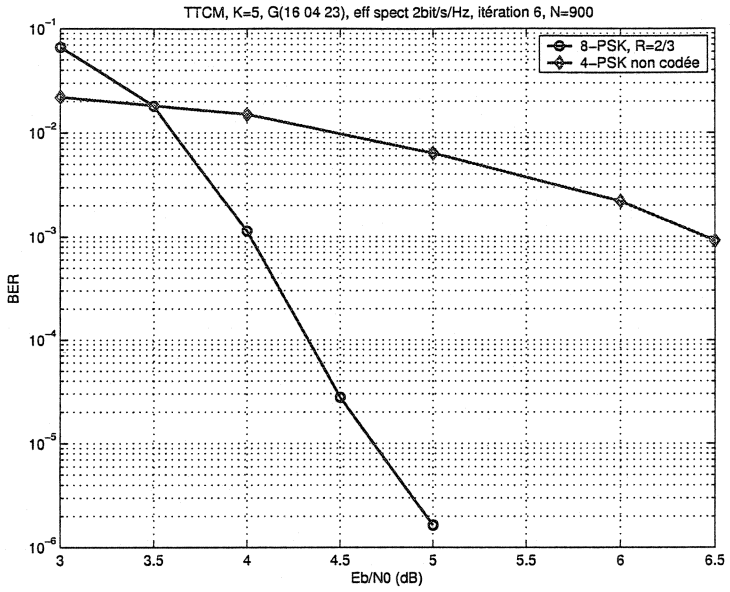

Figure III.2: Gain de codage du schéma TTCM par rapport à la modulation non codée, K=5, G : 16 04 23, N=900, efficacité spectrale 2 bits/s/Hz

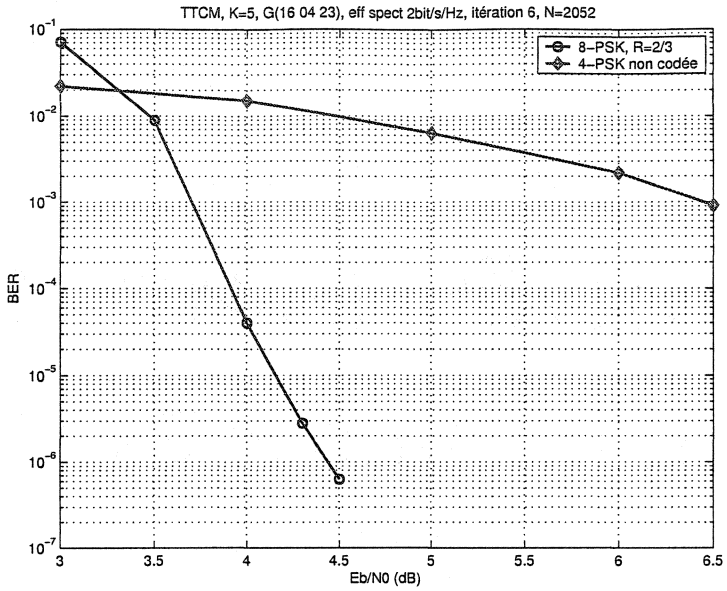

Figure III.3: Gain de codage du schéma TTCM par rapport à la modulation non codée, K=5, G : 16 04 23, N=2052, efficacité spectrale 2 bits/s/Hz

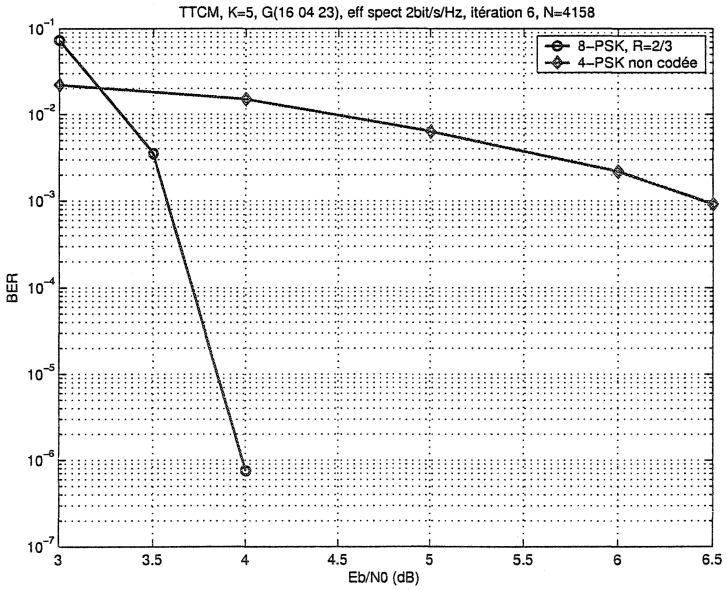

Figure III.4: Gain de codage du schéma TTCM par rapport à la modulation non codée, K=5, G : 16 04 23, N=4158, efficacité spectrale 2 bits/s/Hz

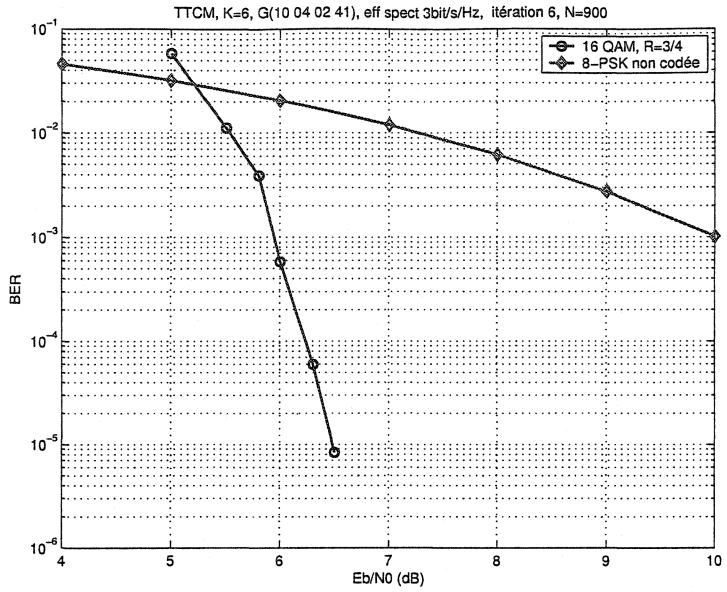

Figure III.5: Gain de codage du schéma TTCM par rapport à la modulation non codée, K=6, G : 10 04 02 41, N=900, efficacité spectrale 3 bits/s/Hz

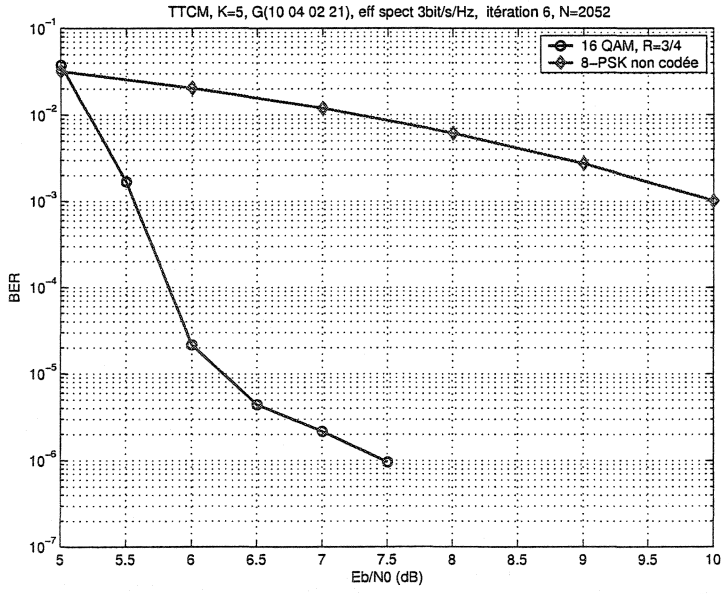

Figure III.6: Gain de codage du schéma TTCM par rapport à la modulation non codée, K=5, G : 10 04 02 21, N=2052, efficacité spectrale 3 bits/s/Hz

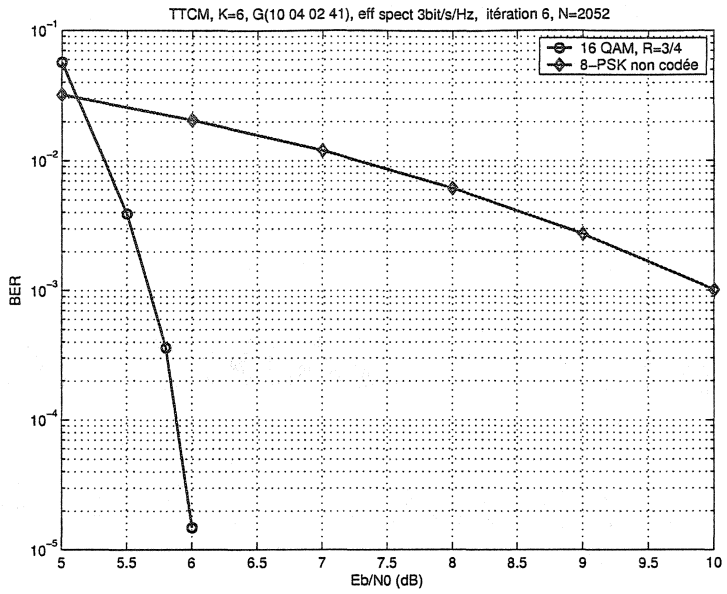

Figure III.7: Gain de codage du schéma TTCM par rapport à la modulation non codée, K=6, G : 10 04 02 41 , N=2052, efficacité spectrale 3 bits/s/Hz

Annexe IV

Courbes comparatives du codeur TTCM et du codeur turbo pragmatique

Nous présentons dans cet annexe les courbes relatives à la comparaison des deux principaux schémas de modulations à savoir les schémas TTCM et turbo pragmatique.

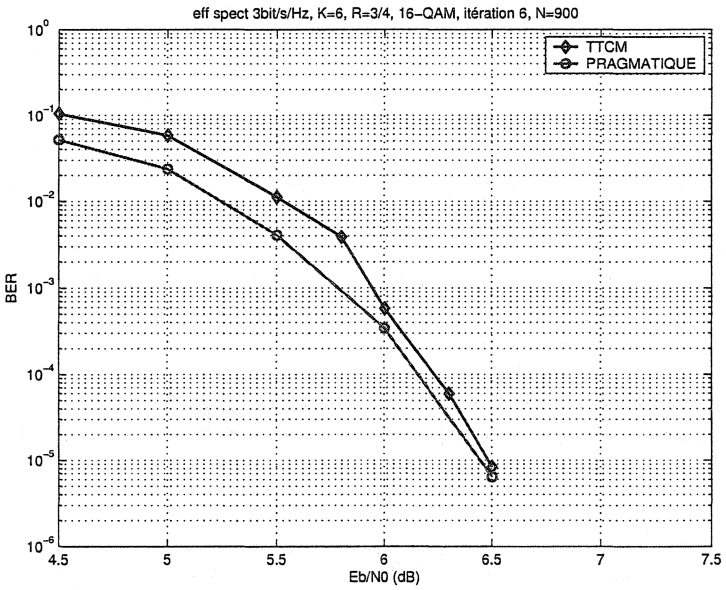

Figure IV.1: Comparaison des schémas TTCM et turbo pragmatique, efficacité spectrale 3 bit/s/Hz, K=6, N=900, 16-QAM

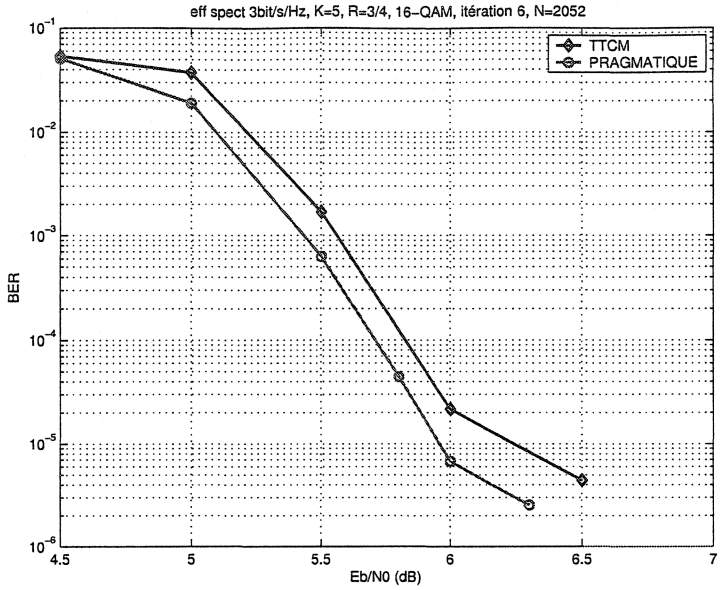

Figure IV.2: Comparaison des schémas TTCM et turbo pragmatique, efficacité spectrale 3 bit/s/Hz, K=5, N=2052, 16-QAM

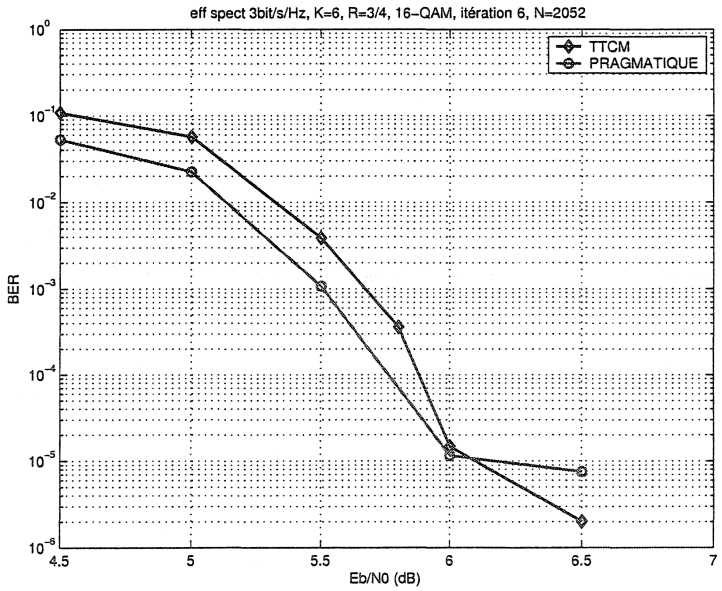

Figure IV.3: Comparaison des schémas TTCM et turbo pragmatique, efficacité spectrale 3 bit/s/Hz, K=6, N=2052, 16-QAM

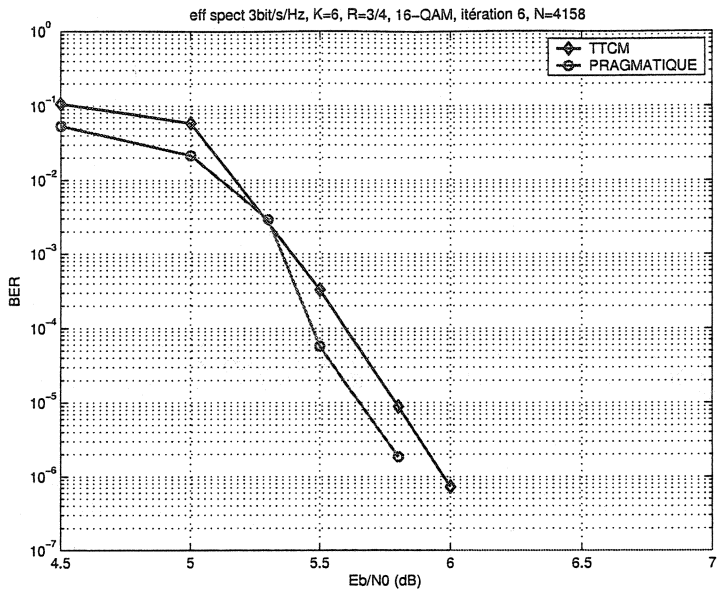

Figure IV.4: Comparaison des schémas TTCM et turbo pragmatique, efficacité spectrale 3 bit/s/Hz, K=6, N=4158, 16-QAM